Leveraging the Potential of MQTT in Industrie 4.0

Florian Pethig
Fraunhofer IOSB, IOSB-INA Lemgo, Fraunhofer Institute of Optronics, System Technologies and Image Exploitation
florian.pethig@iosb-ina.fraunhofer.de

Abstract—**Interoperability and scalability are key requirements for the implementation of Industrie 4.0 (I4.0) application scenarios targeting flexible and resilient production systems. In place of the rigid structures and heterogeneous communication solutions used in Operational Technology (OT) today, loosely coupled systems using standardized interfaces and information models are needed in order to decrease the high manual effort for (re-)configuration. With this regard, the Publish-Subscribe (PubSub) communication paradigm is well suited for the implementation of loosely coupled systems, that can e.g. be implemented using the MQTT application protocol. However, MQTT is data agnostic and does not address interoperability on the information layer, i.e. the definition of topic trees and payloads are up to the user. Specifications like Sparkplug B (Sparkplug) and the standard OPC Unified Architecture (OPC UA) use MQTT as a transport protocol and add such missing features. In this article, Sparkplug and OPC UA are compared according to the requirements of the I4.0 scenarios Plug & Work and Condition Monitoring.**

Index Terms—**Industrie 4.0, MQTT, OPC UA, Sparkplug B**

I. Introduction

In this day and age, increasing the flexibility and resilience of manufacturing faces challenges regarding interoperability and scalability of technical systems [1], [2]. As a key enabler for these two key aspects of Industrie 4.0 (I4.0), the rigid Operational Technology (OT) and Information Technology (IT) architectures widely used today, have to develop into the direction of a dynamic and loosely coupled Industrial Internet of Things (IIoT) [3], [4].

This is mainly due to high manual effort that is needed today to (1) configure point-to-point connections and (2) map data variables between lots of different and changing systems, e.g. Programmable Logic Controllers (PLCs), Manufacturing Execution Systems (MES), and Data Analytics Tools. Additionally, the (3) workload on systems and (4) network infrastructure becomes a bottleneck when trying to scale out the classic architecture, e.g. for large-scale distributed I4.0 application scenarios like (Collaborative) Condition Monitoring or (global) Plant Optimization. A tight coupling works in industrial control setups with a manageable amount of devices, where systems should know (about the state of) each other to achieve certain levels of Quality-of-Service (QoS), e.g. with regard to determinism, reliability, and maintainability [5]. Nevertheless, I4.0 scenarios like "Plug & Work" require a loser coupling as well [3].

Avoiding the issues (1) - (4) during the implementation of the I4.0 application scenarios mentioned above requires a decoupling of the communicating entities in space, time, and synchronization [6]. These entities should not have to know each other (space), they should not have to be active at the same time, and they should not be blocked while producing events or waiting for events (synchronization) [6]. Exactly this can be achieved with the PubSub communication paradigm. Here, messages from publishers are broadcasted to multiple subscribers, who are interested in certain topics or content [6]. These broadcasts are handled by a Message Oriented Middleware (MOM), which is a software or hardware infrastructure that supports sending and receiving messages between distributed systems. Several technologies offer implementations of the PubSub communication paradigm, e.g. Apache Kafka, AMQP or MQTT [7]–[9]. Especially MQTT is achieving increasing attention for IIoT Use Cases and is used as a transport protocol in standards like OPC UA and specifications like Sparkplug B (Sparkplug) [10], [11]. Reasons for this are that MQTT already addresses most of the challenges mentioned above, namely (1) PubSub for loose coupling, (3) focus on a small footprint for embedded devices, and (4) unreliable low bandwidth networks [8]. Additionally, MQTT's protocol design is kept simple and does not specify an information or topic model [11]. But looking at challenge (2) from above, in the IIoT this can turn out to rather be a disadvantage. For this reason, MQTT topic namespaces and payload structures are defined by OPC UA and Sparkplug.

This paper is organized as follows. Section II introduces the I4.0 application scenarios Plug & Work (P&W) and Condition Monitoring (CM) with regard to their requirements. Section III introduces the state of the art for the implementation of IIoT application scenarios by covering the popular technologies MQTT, OPC UA over MQTT and Sparkplug. Section IV analyzes the solutions OPC UA and Sparkplug based on the requirements of the application scenarios from Section II. Section V concludes this paper by summarizing results and identifying open issues for future work.

II. Industrie 4.0 Application Scenarios

This section describes application scenarios addressing two key aspects of Industrie 4.0: Flexibility of production systems (PS) to rapidly adapt to change, and Condition Monitoring (CM) as base requirement for resilience by preventing failures in these PS. Requirements are derived from the application scenarios and merged, if possible.

A. Plug & Work

The application scenario P&W addresses the flexibility and changeability of production systems, i.e. to integrate new components and thus enhanced functionality, modify or update existing entities, or extract existing entities to disable certain functionalities [4], [12]. Today, a lot of manual effort is needed for the (re-)configuration of such hard-coded systems, e.g. to (re-)map variables in IEC 61131-3 PLC programs [13]. This approach is not only time consuming, but also error prone. Main requirements for the application of P&W of PS components are [4]: (r1) Component descriptions: Machine interpretable and uniquely identifiable capability description, (r2) Component selection: Automatic comparison of function-oriented descriptions of production tasks and components' capabilities, (r3) Component access: Uniform interfaces to components for orchestration by higher layer control systems, and (r4) Component control: Modular, self-adapting information and control structures for components.

B. Condition Monitoring

CM describes the process of monitoring the condition of equipment (components, machines, PS) via suitable data variables, e.g. vibration or temperature, particularly for anomaly detection [14]. A CM System (CMS) consists of modules for the collection, analysis and visualization of process and alarm data [15], whereby the analysis again consists of model learning and anomaly detection phases [16]. To date, especially data collection suffers from challenges regarding the variety and quantity of heterogeneous industrial communication systems deployed in the field [2], [17]–[20]. These systems use different interfaces and information models leading to (1) and (2) from Section I. Most of these systems establish point-to-point connections via the Client Server communication paradigm leading to (3) and (4) from Section I, especially for enterprise-wide and global CMS. In this scenario industrial data would be exchanged over the Internet, which opens PS to new threats and automatically leads to certain requirements regarding Cybersecurity [2]. In many brownfield scenarios additional IoT sensors would be installed that might rely on batteries and mobile networks. These devices profit from protocols that only lead to so much workload and need low bandwidth. Hence, the following requirements for the efficient implementation of CM in PS can be derived: (r5) Uniform interface for data acquisition, (r6) Standardized semantic information models, (r7) Scalability, (r8) Cybersecurity, and (r9) Low Bandwidth.

C. Overall Requirements

For the sake of clarity, some requirements from above can be merged. (r3) and (r5) can be merged to **(R1)** Uniform interface, since component access is a prerequisite for data acquisition. Since capabilities, task descriptions, and modular control structures can be contained in information models, (r1), (r2), (r4) and (r6) are merged to **(R2)** Standardized semantic information models. The following requirements for CM remain: **(R3)** Scalability, **(R4)** Low Bandwidth, and **(R5)** Cybersecurity. In general, the capital intensive nature of manufacturing, as well as possible safety threats have to be taken into account as well [21]–[23]. Especially flexible PS should be based on international standards (**R6**) developed and adopted by major companies (**R7**) in order to achieve an acceptable level of security for investment [21], [24]. Additionally, the compliance of products implementing these standards should be tested and certified to assure their reliability **(R8)** [25].

III. State of the Art

A. MQTT

MQTT originated from a use case in which oil pipelines should be connected via satellite connections [26]. Thus, important requirements included minimal bandwidth usage and QoS for data delivery. MQTT is published by the Organization for the Advancement of Structured Information Standards (OASIS) and version 3.1.1 is additionally standardized as ISO/IEC 20922:2016 [27], [28]. MQTT is data agnostic and based on a transport protocol that must provide ordered, lossless, bi-directional connections, e.g. the Transmission Control Protocol (TCP). Using these connections, a protocol for the communication between Clients (Publishers and Subscribers) and a Server (Message Broker) is specified that offers messaging with minimal protocol overhead (2 byte fixed header length), three QoS levels (”at most once”, ”at least once”, ”exactly once”) and notifications about abnormal disconnections. MQTT uses registered port numbers 1883 (mqtt) and 8883 (secure-mqtt) and can be tunneled through port 80 using WebSockets. In MQTT, the Broker is responsible for message filtering, message distribution, authentication and authorization [26]. After a client has established a connection to the Broker, it can publish messages using a PUBLISH control packet, i.a. including a topic name as UTF-8 encoded string consisting of one or more topic levels separated by a forward slash [26]. Clients can subscribe to such topics by sending the corresponding SUBSCRIBE control packet to the Broker, i.a. including one or more single- (”+”) or multi-level (”#”) topic filters. The Broker then distributes messages published to topics to subscribers. MQTT 5 supports additional features that evolved around handling topics, e.g. shared subscriptions, topic aliases, and request-response style communication via corresponding topics [29]. Some well-known Brokers include HiveMQ [30], Eclipse Mosquitto [31] or RabbitMQ [32].

B. OPC UA over MQTT

OPC UA is an industrial interoperability framework and IEC Standard developed by the OPC Foundation (OPCF) [33]. The OPCF is a global organization consisting of over 900 members committed to secured, robust industrial interoperability from sensor to cloud and back independent from any vendor, operating system, market and language [34]. The technology is driven by the member companies sending their experts into the working groups of the OPCF to extend and enhance the specifications. OPC UA can be considered as a modular toolbox for I4.0 compliant communication [35]. The modules of the toolbox correspond to the different parts of the OPC UA standard like transport, security, and information modeling.

1) Transport: For transport, OPC UA supports Client Server and PubSub communication paradigms, that can both be used in parallel [36]. OPC UA PubSub is transported via a MOM, which in its simplest form can be network infrastructure that is able to broadcast datagrams in locally distributed systems. The alternative is using a MOM based on a Message Broker, which is well suited for the large-scale distributed systems mentioned before, where potentially

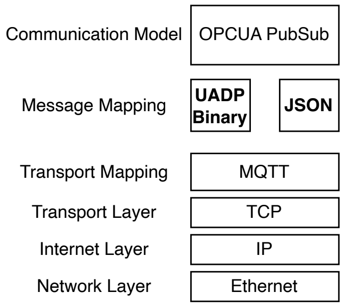

Fig. 1: OPC UA over MQTT Protocol Stack

millions of clients exchange messages via unreliable wide area networks and the resulting Big Data has to be exchanged as a stream of messages with business applications most efficiently. OPC UA PubSub can be transported Broker-less using the User Datagram Protocol (UDP) or directly via Ethernet, which is important for real-time communication on the shopfloor [37]. The Broker-based variant of OPC UA PubSub can be transported via Advanced Message Queuing Protocol (AMQP) or MQTT. AMQP emphasizes transactional mechanisms and is mainly driven by business cases of the financial sector [9]. In this article, the focus is on the OPC UA over MQTT protocol stack depicted in Figure 1 targeting IIoT scenarios, e.g. using the supported Cloud solutions Microsoft Azure and Amazon Web Services (AWS) [38], [39]. OPC UA PubSub messages can be encoded using JSON or a UADP message mapping based on an optimized version of the OPC UA binary encoding [36]. For automatic features like discovery, status and semantic information a standardized MQTT topic tree is necessary and will be included in version v1.05.03 of [36]. Because of industry feedback, which require to allow any topic tree structure for the data messages, this topic tree will not be mandated. Nevertheless, a suggestion will be provided.

2) Security: Security is an integral part of OPC UA and focuses on known threats and countermeasures for industrial automation systems [40]. For OPC UA PubSub different levels of security can be achieved depending on the environment and the chosen encoding scheme. For the Broker-less variant of OPC UA PubSub, confidentiality and integrity of messages can be assured by using symmetric encryption and signature algorithms. When using a broker for OPC UA PubSub communication, often the broker itself offers mechanisms for authorization of clients and to meet confidentiality, integrity and authentication. Additionally, for UADP the same symmetric key concept as defined above can be used to disclose message content from the broker, decreasing the risk of man-in-the-middle attacks. The keys needed for message security can be distributed via a specified Security Key Server and thus do not have to be shared via the MOM [36]. Using OPC UA's JSON encoding, end-to-end message security cannot be assured. Nevertheless, non-repudiation can be implemented and allows subscribers to know the true origin of the data. In general, the Security of OPC UA is reviewed by the German BSI [41].

3) Information Modeling: The basic information model of OPC UA describes standardized nodes of a server's address space. Additionally, predefined base models exist, e.g. for Data Access, Alarms & Conditions, and Historical Access [42]. On top of these basic models other organizations define so called Companion Specifications (CS). These OPC UA CS enable interoperability in and between different domains like plastics, robotics, machine tools, sensors and 85+ more. Especially in machinery and equipment manufacturers invest a lot of effort in the standardization of CS to enable use cases like P&W and CM by reducing the effort for data integration based on semantics included in CS [35]. For example, the German VDMA (Machinery and Equipment Manufacturers Association), representing about 3.500 German and European companies of the mechanical engineering industry, is standardizing CS for machinery enabling cross-sector interoperability. On top of CS, vendor-specific information can always be modeled to account for specific use cases and intellectual property. Focusing on PubSub, parts of an OPC UA application's address space called DataSets can be configured for being published as payload of messages to a MOM [36]. Syntax and semantics of the DataSet are described in DataSetMetaData messages that can also be shared between publishers and subscribers independent of the MOM. This way, it's possible to use the many existing standardized information models in PubSub scenarios efficiently. Publishers and subscribers can be configured by generic configuration tools using the specified PubSub configuration model contained in an OPC UA Server.

C. Sparkplug

The Sparkplug specification has initially been released in 2016 aiming at increasing the interoperability of MQTT based solutions in the IIoT and Supervisory, Control and Data Acquisition (SCADA) market [11]. Since 2019, Sparkplug is an open-source specification hosted at the Eclipse Foundation. The latest version 3.0.0 of the specification has been published in late 2022 [43]. In 2023, the Eclipse Sparkplug Working Group is mainly driven by 12 companies and offering a Technology Compatibility Kit (TCK) and logo program for companies to certify Sparkplug compliance [44], [45]. Sparkplug specifies a topic namespace, state management, and payload definition for the communication between a Host Application and Edge Nodes via a Sparkplug compliant or aware MQTT Server as central Message Broker [43].

1) Topic Namespace: Sparkplug specifies the `spBv1.0` namespace, that enables the unique identification and logical grouping of Edge Nodes and attached devices via corresponding identifiers. A message type part of the topic describes how to interpret the payload and enables a host application to discover metadata, metrics, write commands, and monitor the state of nodes and devices [43].

2) MQTT State Management: Managing the state of components in a Sparkplug infrastructure is implemented around a set of specified *birth* and *death* certificates in conjunction with MQTT *will messages* and the connection *keep alive* timer [43]. Subscribers can be notified if a MQTT client connection

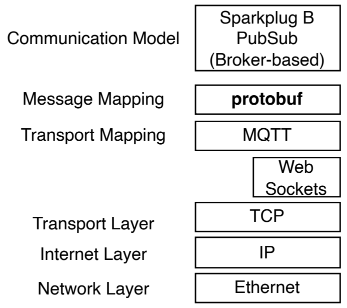

Fig. 2: Sparkplug Protocol Stack

is lost ungracefully, since persistent nodes stay connected at all times allowing the host application to know the state of these nodes based on the keep alive period and the concept for birth and death. Host applications use a specific topic to signal their availability to subscribers. Since time is an important aspect of state management, Sparkplug defines that all timestamps must be in Coordinated Universal Time (UTC) and that participants must have a mechanism to ensure their clocks remain accurate, e.g. based on the Network Time Protocol (NTP). Based on this state management, Sparkplug also supports Report by Exception (RBE) and transmits values only if they changed.

3) MQTT Payload Definition: The original payload definition is called Sparkplug A and has been replaced by Sparkplug B already in version 2.1 of the specification [11]. As depicted in Figure 2, Sparkplug uses protobuf for encoding message payloads. Protobuf is an approach by Google for a language-neutral, platform-neutral extensible mechanism for (de-)serializing structured data to and from a “compact, forward- and backward-compatible binary format” [46]. It uses concepts like `varint` and packing to reduce data size. Protobuf is open source under Apache 2.0 license since 2008. Nevertheless, to date only Google Cloud offers native support for protobuf [47]. Sparkplug defines a protobuf schema consisting of an array of metrics with metadata surrounding those metrics. A metric represents a key, value, timestamp, and datatype along with metadata used to describe the information it contains. Metadata is mainly based on content descriptions in the form of UTF-8 encoded strings, e.g. containing plain text, JSON, or XML. Nevertheless, custom data can be added as an array of bytes as well, e.g. a PDF documentation file. Additionally, a specified *PropertySet* can be used to add custom properties to a metric, e.g. a unit for a certain measurement. Further, templates can be used to define complex datatypes as an array of metrics, which can also be other templates or parameter definitions with default values.

4) Security: Security is a non-normative part of the Sparkplug specification, that provides a guideline on how to secure a Sparkplug infrastructure based on the security sections of MQTT [11], [29]. For Confidentiality, mechanisms for encrypting the underlying connection, e.g. Transport Layer Security (TLS), shall be used. Additionally, Access Control Lists (ACL) are used to restrict publish or subcribe access to topics in the MQTT Broker, that shall be able to decrypt, decode and read all exchanged messages.

IV. Requirements Analysis

In this section the two solutions OPC UA and Sparkplug are analyzed according to the requirements from Section II.

(R1) Uniform Interface

Both solutions aim at increasing industrial interoperability by providing a uniform interface based on MQTT. Sparkplug specifies a MQTT topic namespace for data, discovery and status of host applications, edge nodes, and devices. This topic namespace is fixed for all Sparkplug compliant applications. OPC UA will include a standardized topic tree for discovery, status and semantic information, as well as custom topic structures for data messages. The Sparkplug interface specifies a protobuf schema for binary message mapping natively supported by Google Cloud. In addition to its own binary mapping UADP, OPC UA over MQTT also offers a JSON mapping and is supported by Microsoft Azure and AWS.

(R2) Standardized Semantic Information Models

Establishing standardized semantic information models is a main goal of OPC UA’s CS. There are currently 85+ information models available on the public website, that are being developed together with other organizations, industry associations, and domain experts. CS are available for classes of devices, machines, and complete domains like energy and factory automation. Publishing parts of these CS via MQTT is an inherent feature of OPC UA. Sparkplug focuses on a generic model for SCADA use cases and currently does not specify industry specific information models.

(R3) Scalability

In this article, (R3) has been derived from the application scenario CM in order to address the common challenges from Section I. OPC UA addresses (1) in the sense of maintainability by defining a generic configuration model and topics containing configuration information, allowing to centrally manage, configure and discover an arbitrary number of OPC UA PubSub enabled devices, e.g. using standard OPC UA Clients. Sparkplug currently relies on the configuration of publishers and subscribers on source code level or using device-specific tools. Challenge (2) is addressed by OPC UA’s CS and publishing parts of these as semantic information via corresponding topics. Sparkplug offers to publish metadata as well. Nevertheless, this metadata has to be defined by the user. The workload on systems (3) highly depends on the implementation of a specification. Sparkplug offers reference implementations in C, Java, Javascript, and Python. In addition to numerous reference implementations supporting a variety of programming languages, e.g. C, C#, Java, Javascript, OPC UA specifies implementation profiles, e.g. to scale from a single sensor up to a full-featured OPC UA application in the Cloud. Scalable communication is derived from (4) and a main goal of MQTT and thus addressed by both solutions, e.g. using a clustered Broker infrastructure.

(R4) Low Bandwidth

Using low bandwidth is a key intention of Sparkplug and a reason to use protobuf for binary encoding of messages. OPC UA offers its own binary mapping called UADP. Current work already focuses on a detailed comparison of the UADP and protobuf encodings. For the reasons stated in Section III, OPC UA additionally offers a JSON encoding of messages. For good reasons described in Section III-B, this encoding will certainly consume more bandwidth than binary encodings.

(R5) Cybersecurity

Both solutions offer to use TLS and MQTT security features. In addition, OPC UA over MQTT allows for application-to-application integrity and confidentiality, if UADP is used. This allows scenarios, where intermediate brokers will not be able to read the data. OPC UA over MQTT with the JSON encoding will offer non-repudiation which allows subscribers to know the true origin of the data, which is an important requirement for many applications. OPC UA Security has been reviewed by German BSI and other international security experts due to relevance to their industry.

(R6) International Standard

OPC UA is published as IEC standard 62541. Sparkplug is published as open specification by the Eclipse foundation.

(R7) Adoption

As listed in Section III-B, the OPCF consists of over 900 members. OPC UA over MQTT is currently adopted by 26 companies from OT, as well as AWS and Azure as major Cloud solutions. These companies participate in a designated plugfest activity since 2021 [48]. Sparkplug currently lists 12 companies developing the specification on the website (see Section III-C). No specific coverage about Sparkplug's adoption could be found during this research.

(R8) Compliance & Certification

OPCF runs a compliance and certification program which includes interoperability testing between different vendors. A Conformance Test Tool (CTT) is available including over 1.800 test scripts for OPC UA and its CS. Certification is optional for OPC enabled products, but all certified applications are required to go through a third-party assessment in an OPCF Certification Test Lab. The certification and compliance testing is performed both on the core OPC UA specifications and on industry defined CS. HiveMQ provides a Technology Compatibility Kit (TCK) for Sparkplug 3.0, that can be used to certify conformance to profiles for Host Applications, Edge Nodes and MQTT Brokers. This includes 6 tests for Host Applications, 5 for Edge Nodes, as well as tests for Sparkplug compliant or Sparkplug aware (including state management) MQTT Brokers. Sparkplug offers a logo program for products that have been certified compatible.

V. Summary and Future Work

This paper introduced challenges as well as specific requirements for the implementation of the two I4.0 application scenarios P&W and CM. A uniform interface (**R1**) is needed and provided by both solutions on top of MQTT. With regard to defined fixed topic namespaces, OPC UA plans to stay more flexible by additionally allowing custom topic trees for data messages and providing semantic information via specified topics. While both solutions offer efficient binary encodings, OPC UA also includes a JSON message mapping, that can be understood by humans without special tooling and applications unaware of OPC UA. Standardized semantic information models (**R2**) are crucial for the implementation of P&W (see Section II) but also facilitate data integration for CM. Here, OPC UA has a lot to offer with its CS that can be transported via OPC UA over MQTT as well. A scalable (**R3**) network infrastructure is addressed by both solutions. Nevertheless, especially OT solutions must be (re-)configured regularly and thus profit from OPC UA's configuration model and topic structure. Both solutions offer efficient binary encodings for low bandwidth consumption (**R4**). Regarding Cybersecurity (**R5**), OPC UA over MQTT can disclose message content from the Broker and assure the true origin of data. Here, Sparkplug solely relies on MQTT's Cybersecurity features. While Sparkplug is an open specification published by the Eclipse Foundation, OPC UA is an established IEC Standard (**R6**). The stable IEC workflow is often recognized as high value by organizations, that e.g. build their information models on top of OPC UA. OPC UA over MQTT is adopted (**R7**) by major companies from IT and OT that work on the interoperability of their solutions for already two years now. OPC UA runs a well established compliance and certification program (**R8**), while the TCK for Sparkplug compliance testing is offered by the company HiveMQ.

In conclusion, Sparkplug adds important features for the implementation of I4.0 application scenarios to MQTT. It can be reasonable to use Sparkplug, if a system's configuration does not change too often and if a binary encoding is what is needed. Especially the application scenario P&W requires efficient (re-)configuration of PS potentially based on standardized semantic information models (see overall requirements in Section II). In this regard, OPC UA offers advantages like CS and a generic configuration model. Additionally, OPC UA being an international standard adopted by major companies is an advantage for the security of investment (see Section II).

Future work should compare the binary encodings UADP and protobuf in depth and also evaluate the usage of Sparkplug and OPC UA in a real world setting, e.g. actual measurements for latency and bandwidth could provide more insights. Additionally, mapping OPC UA's CS to Sparkplug should be evaluated. Furthermore, the workload of Sparkplug and OPC UA implementations needs to be analyzed for different scenarios, e.g. from a sensor publishing a single value to applications aggregating global plant data in the Cloud.

References

[1] H. Panetto, B. Iung, D. Ivanov, G. Weichhart, and X. Wang, "Challenges for the cyber-physical manufacturing enterprises of the future," *Annual Reviews in Control*, vol. 47, pp. 200 – 213, 2019.

[2] K. Al-Gumaei, A. Müller, J. N. Weskamp, C. S. Longo, F. Pethig, and S. Windmann, "Scalable analytics platform for machine learning in smart production systems," in *24th IEEE ETFA*, 2019, pp. 1155–1162. [Online]. Available: https://dx.doi.org/10.1109/ETFA.2019.8869075

[3] P. Leitão, A. W. Colombo, and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," *Computers in Industry*, vol. 81, pp. 11–25, 2016, emerging ICT concepts for smart, safe and sustainable industrial systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166361515300348

[4] M. Schleipen, A. Lüder, O. Sauer, H. Flatt, and J. Jasperneite, "Requirements and concept for plug-and-work," *at – Automatisierungstechnik*, vol. 63(10), pp. 801–820, 2015.

[5] M. Riedl, H. Zipper, M. Meier, and C. Diedrich, "Cyber-physical systems alter automation architectures," *Annual Reviews in Control*, vol. 38, no. 1, pp. 123–133, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578814000133

[6] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, p. 114–131, jun 2003. [Online]. Available: https://doi.org/10.1145/857076.857078

[7] Apache Kafka. [Online]. Available: https://kafka.apache.org/

[8] MQTT. [Online]. Available: https://mqtt.org/mqtt-specification/

[9] AMQP - Advanced Message Queuing Protocol. [Online]. Available: https://www.amqp.org

[10] OPC Foundation - List of Specification Documents. [Online]. Available: https://opcfoundation.org/developer-tools/documents/?type=Specification

[11] The Sparkplug Specification. [Online]. Available: https://sparkplug.eclipse.org/specification/

[12] J. Jasperneite, S. Hinrichsen, and O. Niggemann, "Plug-and-produce für fertigungssysteme," *Informatik Spektrum*, vol. 38-3, pp. 183–190, 2015.

[13] F. Pethig, O. Niggemann, and A. Walter, "Towards Industrie 4.0 compliant configuration of condition monitoring services," in *15th IEEE INDIN*, 2017, pp. 271–276.

[14] J. Eickmeyer, P. Li, O. Givehchi, F. Pethig, and O. Niggemann, "Data driven modeling for system-level condition monitoring on wind power plants." in *DX*, 2015, pp. 43–50.

[15] M. Fullen, P. Schüller, and O. Niggemann, "Defining and validating similarity measures for industrial alarm flood analysis," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 2017, pp. 781–786.

[16] S. S. Gilani, S. Windmann, F. Pethig, B. Kroll, and O. Niggemann, "The importance of model-learning for the analysis of the energy consumption of production plants," in *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2013, pp. 1–8.

[17] F. Pethig, B. Kroll, O. Niggemann, A. Maier, T. Tack, and M. Maag, "A generic synchronized data acquisition solution for distributed automation systems," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*. IEEE, 2012, pp. 1–8.

[18] S. Faltinski, H. Flatt, F. Pethig, B. Kroll, A. Vodenčarević, A. Maier, and O. Niggemann, "Detecting anomalous energy consumptions in distributed manufacturing systems," in *IEEE 10th International Conference on Industrial Informatics*, 2012, pp. 358–363.

[19] K. Al-Gumaei, K. Schuba, A. Friesen, S. Heymann, C. Pieper, F. Pethig, and S. Schriegel, "A survey of internet of things and big data integrated solutions for industrie 4.0," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2018, pp. 1417–1424.

[20] E. Trunzer and F. Pethig, "Systemarchitekturen für Smart Data Ansätze Aggregiertes Konzept aus mehreren Projekten," *Produktions- und Verfügbarkeits-optimierung mit Smart Data Ansätzen*, p. 13, 2018.

[21] L. M. Tosatti, "Life cycle cost calculation for investment decision in manufacturing," in *13th CIRP International Conference on Life Cycle Engineering, Proceedings of LCE2006*. Citeseer, 2006, pp. 723–7.

[22] A. Klose, F. Pelzer, M. Barth, R. Drath, R. Oehlert, S. V. León, A. Horch, C. Kotsch, J. Knab, B. Gut, H. Manske, and L. Urbas, "Distributed functional safety for modular process plants," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1381–1384.

[23] T. Schulz, C. Griest, F. Golatowski, and D. Timmermann, "Strategy for security certification of high assurance industrial automation and control systems," in *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)*, 2018, pp. 1–4.

[24] M. Fechter, T. Dietz, and T. Bauernhansl, "Cost calculation model for reconfigurable, hybrid assembly systems," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, 2019, pp. 836–841.

[25] F.-W. Jaekel, T. Wolff, V. Happersberger, and T. Knothe, "Ensure opc-ua interfaces for digital plug-and-produce," in *On the Move to Meaningful Internet Systems: OTM 2019 Workshops*, C. Debruyne, H. Panetto, W. Guédria, P. Bollen, I. Ciuciu, G. Karabatis, and R. Meersman, Eds. Cham: Springer International Publishing, 2020, pp. 44–53.

[26] N. F. Syed, "Iot-mqtt based denial of service attack modelling and detection," Ph.D. dissertation, Edith Cowan University, 2020.

[27] MQTT Version 3.1.1. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[28] ISO/IEC 20922:2016 Information technology — Message Queuing Telemetry Transport (MQTT) v3.1.1. [Online]. Available: https://www.iso.org/standard/69466.html

[29] MQTT Version 5.0. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[30] HiveMQ. [Online]. Available: https://www.hivemq.com/

[31] Eclipse Mosquitto - An open source MQTT broker. [Online]. Available: https://mosquitto.org/

[32] RabbitMQ. [Online]. Available: https://www.rabbitmq.com/

[33] "IEC TR 62541-1:2020 OPC Unified Architecture - Part 1: Overview and concepts," International Electrotechnical Commission, Standard, Nov. 2020.

[34] Members of the OPC Foundation. [Online]. Available: https://opcfoundation.org/members

[35] VDMA (Machinery and Equipment Manufacturers Association), Ed., *Industrie 4.0 Communication Guideline Based on OPC UA*. VDMA Verlag GmbH, 2017. [Online]. Available: https://www.researchgate.net/publication/320035806_Industrie_40_Communication_Guideline_Based_on_OPC_UA

[36] "OPC 10000-14 UA Part 14: PubSub V1.05.02," OPC Foundation, Scottsdale, US, Specification, Nov. 2022.

[37] "OPC Unified Architecture Field eXchange (UAFX)," OPC Foundation, Scottsdale, US, Specification, Nov. 2022.

[38] What is OPC Publisher? [Online]. Available: https://learn.microsoft.com/de-de/azure/industrial-iot/overview-what-is-opc-publisher

[39] IoT SiteWise OPC-UA collector. [Online]. Available: https://docs.aws.amazon.com/greengrass/v2/developerguide/iotsitewise-opcua-collector-component.html

[40] "OPC Unified Architecture Specification - Part 2: Security Model Release 1.04," OPC Foundation, Standard, Aug. 2018.

[41] "Sicherheitsanalyse OPC UA," Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, DE, Security Analysis, Jun. 2022.

[42] "IEC TR 62541-5:2020 OPC Unified Architecture - Part 5: Information Model," International Electrotechnical Commission, Standard, Jul. 2020.

[43] Open Source Sparkplug Specification on Github. [Online]. Available: https://github.com/eclipse-sparkplug/sparkplug

[44] Eclipse Sparkplug Membership Prospectus. [Online]. Available: https://f.hubspotusercontent10.net/hubfs/5413615/sparkplug-member-prospectus%202020.pdf

[45] A Step-by-Step Guide to Obtaining Sparkplug 3.0 Certification. [Online]. Available: https://www.hivemq.com/article/starter-guide-sparkplug-3-0-certification-technology-compatibility-kit-tck-industry-40/

[46] Protocol Buffers Documentation. [Online]. Available: https://protobuf.dev/

[47] Cloud APIs - Protocol Buffers Version 3. [Online]. Available: https://cloud.google.com/apis/design/proto3?hl=de

[48] SPS Magazin - OPC UA für IT-Konzepte. [Online]. Available: https://www.sps-magazin.de/protokolle-standards/opc-ua-fuer-it-konzepte/