Object-Oriented Internet
Cloud Interoperability

Mariusz Postół[1][0000−0002−9669−0565] and Piotr Szymczak[1][0000−0001−6790−3878]

Institute of Information Technology, Lodz University of Technology, Łódź, Poland
mailto:mariusz.postol@p.lodz.pl

Abstract. Optimization of industrial processes requires further research on the integration of machine-centric systems with human-centric cloud-based services in the context of new emerging disciplines, namely the fourth industrial revolution coined as Industry 4.0 and Industrial Internet of Things. The following research aims at working out a new generic architecture and deployment scenario applicable to that integration. A reactive interoperability relationship of the communication parties is proposed to deal with the network traffic propagation asymmetry or assets' mobility. Described solution based on the OPC Unified Architecture international standard relaxes issues related to the real-time multi-vendor environment. The discussion concludes that the embedded gateway software component best suits all requirements and thus has been implemented as a composable part of the selected reactive OPC UA framework which promotes separation of concerns and reusability.

The proposals are backed by proof-of-concept reference implementations confirming the possibility of integrating selected cloud services with the OPC UA based cyber-physical system by applying the proposed architecture and deployment scenario. It is contrary to interconnecting cloud services with the selected OPC UA Server limiting the PubSub role to data export only.

Keywords: Industry 4.0 · Internet of Things · Object-Oriented Internet · Cloud Computing · Industrial communication · Reactive networking · Machine to Machine Communication · OPC Unified Architecture · Azure

1 Introduction

All the time, Information and Communication Technology is providing society with a vast variety of new distributed applications aimed at micro and macro optimization of the industrial processes. The design foundation of this kind of application must focus primarily on communication technologies. Based on the role humans take while using those applications they can be grouped as follows:

- **human-centric** - information origin or ultimate information destination is an operator,
- **machine-centric** - information creation, consumption, networking, and processing are achieved entirely without human interaction.

A typical **human-centric** approach is a web-service supporting, for example, a web user interface (UI) to monitor conditions and manage millions of devices and their data in a typical cloud-based IoT approach [3, 9, 13, 29]. In this case, it is characteristic that any uncertainty and necessity to make a decision can be relaxed by human interaction. Coordination of robot behaviors in a work-cell (automation islands) is a **machine-centric** example. In this case, any human interaction must be recognized as impractical or even impossible. The interconnection scenario requires machine to machine communication (M2M) [10, 12, 22, 25, 28] demanding the integration of multi-vendor devices.

From the M2M communication concept, a broader idea of a smart factory can be derived. In this M2M deployment approach, the mentioned robots are only executive assets of an integrated supervisory control system responsible for macro optimization of an industrial process composed as one whole. Deployment of the smart factory concept requires a hybrid solution and interconnection of the above mentioned heterogeneous environments. This approach is called the fourth industrial revolution and was coined as Industry 4.0. It is worth stressing that interconnection of machines - or more general assets - is not enough, and additionally, assets interoperability must be expected for the deployment of this concept. In this case, multi-vendor integration makes communication standardization especially important, namely, it is required that the payload of the message is standardized to be factored on the data-gathering site and consumed on the ultimate destination site.

Highly-distributed solutions used to control any real-time process aggregating islands of automation (e.g. virtual power plants producing renewable energy) must, additionally, leverage public communication infrastructure, namely the Internet. The Internet is a demanding environment for highly distributed process control applications designed atop the M2M communication paradigm because it is globally shareable and can be also used by malicious users. Furthermore, it offers only non-deterministic communication making the integration of islands of automation designed against the real-time requirements a demanding task.

Today both obstacles can be overcome, and as examples, we have bank account remote control and voice over IP in daily use. The first application must be fine-tuned in the context of data security, and the second is very sensitive concerning time constraints. Similar approaches could be applied to adopt the concepts well known in process control industry, namely Human Machine Interface (HMI), Supervisory Control and Data Acquisition (SCADA), and Distributed Control Systems (DCS). It is worth stressing that, by design, all of them are designed based on interactive communication. Interactive communication is based on a data polling relationship. If that is the case, the application must follow the interactive behavioral model, because it actively polls the data source for more information by pulling data from a sequence that represents the process state in time. The application is active in the data retrieval process - it controls the pace of the retrieval by sending the requests at its convenience. After dynamically attaching a new island of automation the control application (responsible for the data pulling) must be reconfigured for this interoperability

scenario. In other words the interactive communication relationship cannot be directly applied because the control application must be informed on how to pull data from a new source. As a result, a plug and produce scenario [16] cannot be seamlessly applied. A similar drawback must be overcome if for security reasons suitable protection methods have been applied to make network traffic propagation asymmetric. It is accomplished using intermediary devices, for example, firewalls, to enforce traffic selective availability based on predetermined security rules against unauthorized access.

Going further, we shall assume that the islands of automation are mobile, e.g. autonomous cars passing a supervisory controlled service area. Here, the behavior of the interconnected assets is particularly important concerning the environment in which they must interact. This way we have entered the Internet of Things domain of Internet-based applications.

If we must bother with the network traffic propagation asymmetry or mobility of the asset network attachment-points the reactive relationship could relax the problems encountered while the interactive approach is applied [25]. In this case, the sessionless publisher-subscriber communication relationship is a typical pattern to implement the reactive interoperability paradigm. The sessionless relationship is a message distribution scenario where senders of messages, called publishers, do not send them directly to specific receivers, called subscribers, but instead, categorize the published messages into topics without knowledge about which subscribers, if any, there may be. Similarly, subscribers express interest in one or more topics and only receive messages that are of interest, without knowledge about which publishers, if any, there are. In this scenario, the publishers and subscribers are loosely coupled, i.e. they are decoupled in time, space and synchronization [6].

If the **machine-centric** Cyber-Physical System (CPS) - making up islands of automation - must be monitored and/or controlled by a supervisory system, the cloud computing concept may be recognized as a beneficial solution to replace or expand the above mentioned applications, i.e. HMI, SCADA, DCS, etc. Cloud computing is a method to provide the requested functionality as a set of services. There are many examples that cloud computing is useful to reduce costs and increase robustness. It is also valuable in case the process data must be exposed to many stakeholders. Following this idea and offering control systems as a service, there is required a mechanism created on the service concept and supporting abstraction and virtualization - two main pillars of the cloud computing paradigm. In the cloud computing concept, virtualization is recognized as the possibility of sharing the services by many users, and abstraction hides implementation details.

This article addresses further research on the integration of the multi-vendor **machine-centric** CPS designed atop of M2M communication and emerging cloud computing as a **human-centric** front-end in the context of the Industry 4.0 (I4.0) and Industrial Internet of Things (IIoT) disciplines. For this integration, a new architecture is proposed to support the reactive relationship of communicating parties. To support the multi-vendor environment the OPC Unified

Architecture [7, 17, 18] interoperability standard has been selected. The proposals are backed by proof of concept reference implementations - the outcome has been just published on GitHub as an open-source (MIT licensed) [24]. Prototyping addresses Microsoft Azure Cloud [4] as an example. The proposed solutions have been harmonized with the more general concept called the Object-Oriented Internet (OOI) [20, 23, 24].

The main goal of this article is to provide proof that:

- reactive M2M interoperability based on the OPC UA standard can be implemented as a powerful standalone library without dependency on the Client/Server session-oriented archetype,
- cloud interoperability can be implemented as an external part employing out-of-band communication without dependency on the OPC UA implementation,
- the proposed generic architecture allows that the gateway functionality is implemented as a composable part at run-time - no programming required.

The remainder of this paper is structured as follows. Sect. 2 presents the proposed open and reusable software model. It promotes a reactive interoperability pattern and a generic approach to establishing interoperability-context. A reference implementation of this archetype is described in Sect. 3. The most important findings and future work are summarized in Sect. 4.

2   Sensors to Cloud Interconnection - Architecture

To follow the Industry 4.0 concept a hybrid environment integrating reactive Machine to Machine interconnection and the interactive web-based user interface is required (Sect. 1). The main challenge of the solution in concern is to design a generic but reusable architecture that addresses interoperability of those diverse interconnection scenarios ruled by different requirements, namely:

- **machine-centric** machine to machine real-time mobile interoperability
- **human-centric** cloud-based front-end

Interconnection of the reactive **machine-centric** and interactive **human-centric** environments can be implemented by applying one of the following scenarios:

- **direct interconnection** (tightly coupled)-using a common protocol stack
- **gateway based interconnection** (loosely coupled)-using an out-of-bound protocol stack

By design, the **direct interconnection** approach requires that the cloud has to be compliant with the interoperability standard the CPS uses. As a result, it becomes a consistent communication node of the CPS. The decision to follow the **direct interconnection** scenario must be derived from an analysis of the capabilities of available services in concern. However, for the development strategy of

this type of solution, the analysis can be done partially taking into account the following features that can be considered invariable. By design, the cloud-based services must be virtual - they are used to handle many solutions at the same time. Furthermore, M2M communication is usually constrained by real-time requirements. The virtualization of cloud services means that they must be very flexible to handle the attachment of new assets proactively (acting in advance) at run time. As a result, the cloud services must be responsible to register and authenticate devices by exposing endpoints in the public network to allow the device to access a provisioning cloud service. It requires that a session over the Internet has to be established by the data holding asset at a preparation step. To meet the requirements of real-time distributed control the CPS may use protocols applicable only to local computer networks (e.g. multicast IP, Ethernet, TSN [1], etc.). Because the cloud services support only protocols handling interconnection over the Internet the **direct interconnection** cannot be applied in a general case.

To support also a local network attachment point, the interaction with the cloud requires remote agents implemented by applying one of the following archetypes:

- **edge device** - a remote cloud agent acting as an intermediary for nodes of the CPS
- **field level gateway** - a dedicated custom agent acting as an intermediary for nodes of the CPS
- *Embedded Gateway* - a software part composed into a selected node of the *Cyber-physical network* (Fig. 1)

Edge device connects directly to the cloud services but acts as an intermediary for other devices called leaf devices. Additionally, it allows the selection of initial data and their processing using local resources. The **edge device** may be located close to the leaf devices and attached to the *Cyber-physical network* using protocols applicable only to local computer networks. In this scenario, it is possible to use a custom protocol stack to get connected to the **edge device** with the cloud and to help to save the bandwidth thanks to sending only the results of local processing. In this approach, the **edge device** is part of cloud vendor products and cannot be recognized as a generic solution that can be used to connect to other clouds supporting a many-to-many relationship.

The **field level gateway** is also built atop of the middleware concept [27]. The only difference as compared with the **edge device** is a necessity to use services officially supported by the cloud vendor to get connected. In this scenario, the process data may be transferred to many clouds simultaneously.

Proposition 1. *Unlike the above-described solutions, the* Embedded Gateway *is not derived from the middleware concept. A generic domain model for this interconnection is presented in the Fig. 1. Promoting separation of concerns design principle, the gateway functionality should be implemented as a self-contained*

---

[1] Time-Sensitive Networking (TSN) Task Group https://1.ieee802.org/tsn/

software part embedded in the Networking *service of the* Cyber-physical node*. The main functionality of this component is to transfer selected data between* Cyber-physical network *using* Networking *services of an existing* Cyber-physical node *and* Cloud-based front-end *using interconnection services officially supported by the cloud vendor.*

The interconnection of assets is not enough hence their interoperability is expected. In this case, using the same communication stack must be recognized as only a necessary condition. To support interoperability, common data understanding is required. Additionally, to meet this requirement, the cloud and CPS have to establish directly the same semantic-context and security-context. The possibility of establishing a common semantic-context in the multi-vendor environment makes communication standardization especially important. If that is the case, it is required that the encoding of the payload exchanged over the network (Data Transfer Object) is standardized so that the appropriate messages can be factored on the data-gathering site and consumed on the ultimate destination data processing sites. Security between the data origin and ultimate data destination refers to the protection of messages (security-context) against malicious users. It is required that communicating parties are using the same cyber-security measures. To comply with the Industry 4.0 communication criterion, it is required that any product must be addressable over the network via TCP/UDP or IP and has to support the OPC UA Information Model [15,19,21]. As a result, any product being advertised as Industry 4.0 enabled must be OPC UA-capable somehow.

Fig. 1. Generic interconnection concept

The OPC Unified Architecture interoperability standard has been selected to support the multi-vendor environment. OPC UA supports the following two patterns to be used to transfer data between communicating parties:

- **session-oriented**: requires a session that must be established before any data can be sent between sender and receiver
- **sessionless-oriented**: the sender may start sending messages (called packets or datagrams) to the destination without any preceding handshake procedure

Using the session-oriented communication pattern it is difficult or even impossible to gather and process mobile data (Sec. 1), which is one of the Internet
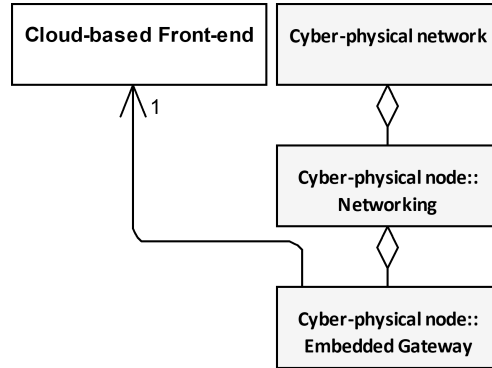
of Things paradigms. OPC UA Part 14 PubSub [2, 26] offers the sessionless approach as an additional option to session-based client-server interoperability relationship and is a consistent part of the OPC UA specifications suit. As a result, it can be recognized as the IoT ready technology.

The proposals presented in the article are backed by proof of concept reference implementations [24]. For this study, prototyping addresses Microsoft Azure cloud products. There are many reasons for selecting Azure to accomplish the cloud-based front-end of a Cyber-Physical System (CPS). Azure offers Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) capabilities. As a result, the platform can be used not only as a cloud-based front-end for CPS. Azure aids Internet protocols and open standards such as JSON, XML, SOAP, REST, MQTT [5], AMQP [8], and HTTP. Software development kits for C#, Java, PHP, and Ruby are available for custom applications.

Based on the sessionless and session-oriented communication patterns examination against the IoT requirements [25] it could be concluded that the connectionless pattern better suites issues related to assets mobility and traffic asymmetry that is characteristic for the application domains in concern. Additionally, to promote interoperability and address the demands of the M2M communication in the context of a multi-vendor environment, the prototyping should use a framework that must be compliant with the OPC UA Part 14 PubSub specification. According to proposed generic architecture (Fig. 1) to implement the *Embedded Gateway* as a composable part of the *Cyber-physical node* a library implementing *Networking* functionality in compliance with above mentioned specification is a starting point for further development. Additionally, it must be assumed that the library used to deploy *Embedded Gateway* supports dependency injection and is capable of composing an external part supporting cloud/PubSub gateway functionality. The composition process must be available without modification of the core code of an existing library. As a result, the prototyping is to be limited to implementation of the *Embedded Gateway* software part only.

To promote interoperability and address the demands of the M2M communication in the context of a multi-vendor environment the prototyping should use a framework that must be compliant with the OPC UA Part 14 PubSub and support the *Reactive Interoperability* (Sect. 1) concept. A framework compliant with these requirements has been implemented as an open-source library[2] named *UAOOI.Networking* (*Networking* for short) under an umbrella of the Object-Oriented Internet project [24]. The library is designed to be a foundation for developing application programs that are taking part in a message-centric communication pattern and interconnected using the reactive networking concept. The diagram in Fig. 2 shows the relationship between the library (*SDK*) and external parts composing any reactive networking application (*Reactive Application*). The *Reactive Application* is an aggregation of parts implementing the *Producer* and *Consumer* roles. By design, they support access to real-time process data, hence they are recognized as an extension of *DataRepository* class. To

---

[2] https://github.com/mpostol/OPC-UA-OOI

implement the *DataRepository* dedicated implementation of the *IBindingFactory* interface should be provided to create a bridge between CPS and an external raw data represented by the *LocalResources* class. A more in-depth description of the *OOI Reactive Application* library enabling data exchange over a network using the reactive networking pattern is covered in [25].

To promote the polymorphic approach, the library has a factory class called *DataManagementSetup* that is a placeholder to gather all injection points used to compose external parts. To be injected, the part supporting data exchange with the underlying process must be compliant with the *IBindingFactory* interfaces. It is expected that the functionality implementation expressed by this interface is provided as an independent external composable part. The composition is accomplished at run time, and the effective application functionality depends essentially on reusable loosely coupled parts composed applying the dependency injection software engineering concept.
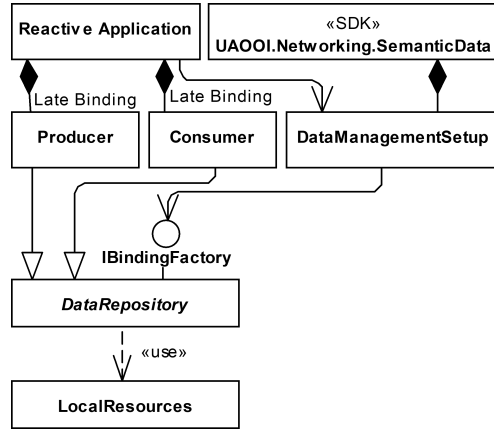
Fig. 2. Reactive interoperability architecture

The *DataRepository* represents data holding assets in the *Reactive Application* implementing the *IBindingFactory* interface. It captures functionality responsible for accessing the external process data from *LocalResources*. The *LocalResources* represents an external part that has a very broad usage purpose. For example, it may be any kind of the process data source/destination, i.e. raw data (e.g. PLC internal registers), OPC UA Address Space Management [25], cloud, file, database, graphical user interface, to name only a few.

Depending on the expected network role the library supports the implementation of:

- *Consumer* - entities processing data from incoming messages,
- *Producer* - entities gathering process data and populating outgoing messages.

The *Consumer* and *Producer* classes are derived from the *DataRepository* (Fig. 2). The *Consumer* uses the *IBindingFactory* to gather the data recovered from the *Message* instances pulled from a network. The received data may be processed or driven to any data destination, e.g. cloud-based front-end. The *Producer* mirrors the *Consumer* functionality and, after reading data from an associated source, populates the *Message* using the gathered data. By design, the *DataRepository* and associated entities, i.e. *Local Resources*, *Consumer*, *Producer* are embedded in external parts, and, consequently, the application scope

may cover practically any concern that can be separated from the core *Reactive Application* implementation.

3   Cloud - OOI Interoperability Implementation

Proposition 2. *A generic domain model presenting interconnection architecture between the* Cloud-based Front-end *and* Cyber-physical node *attached to the* Cyber-physical network *is presented in Fig. 1. It is proposed to implement the* Cyber-physical node *by adopting the* Reactive Application *archetype compliant with the reactive interoperability concept (Fig. 2). Merging selected entities of this archetype into the proposed domain model (Fig. 1) leads to a model expressed as the diagram presented in Fig. 3. In the proposed approach the* Embedded Gateway *is derived from the* Consumer *role implemented as a composable part aggregated by the* Reactive Application*.*

In the final deployment architecture (Fig. 4) the *Consumer* role has been realized by the *PartDataManagementSetup* that is derived from *DataManagementSetup* provided by the library. *Networking* (SDK) was removed from this diagram for the sake of simplicity. Instantiating *PartDataManagementSetup* is the first step for bootstrapping process of the *Consumer* role functionality. This class provides an entry point to initialize all properties, which are injection points of all parts composing this role. It extends the functionality of the *DataManagementSetup* based on the following associated classes: *PartBindingFactory* and *CommunicationContext*.

Fig. 3. Architecture Domain Model

The *PartBindingFactory* implements the *IBindingFactory* to gather the data recovered from the *Message* instances pulled from a network. The received data is driven to *CommunicationContext* for further encoding and finally pushing it to the cloud services using the configured out-of-band protocol.

The cloud interconnection is realized using the *CommunicationContext*. It implements a message encoder and establishes the out-of-band communication stack.
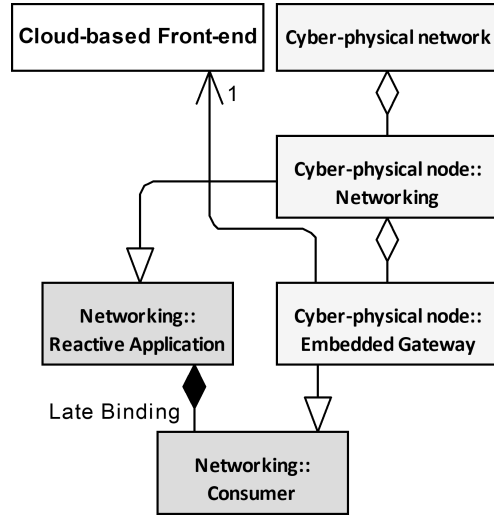
The data recovered from the *Message* is obtained from the *PartBindingFactory* using the *IDTOProvider* that defines a contract used to pull the Data Transfer Object created from a subscription by the *PartBindingFactory*. The transfer process requires data conversion from source to destination encoding, i.e. replacing bitstreams used by the CPS with equivalent ones for the cloud-based services. The Azure offers a vast variety of built-in types ready to be used in common cases, but not necessarily there are equivalent counterparts in use by the CPS. The Azure uses JSON based

Fig. 4. Implementation Architecture

Data Transfer Object encoding and schema defined based on the solution metadata. The PubSub uses JSON and binary Data Transfer Object encodings. In any case, the data recovered from the Message pulled from a subscription is stored locally using the object model based on standard .NET types. *PartBindingFactory* maps selected object graph onto the JSON message required by the cloud services.

The encoded JSON messages must be transferred to cloud over the network using the selected protocol stack. The Azure supports HTTP, AMQP, and MQTT protocol stacks, which are all standard ones. Consequently, it is possible to apply any available implementation compliant with an appropriate specification to achieve connectivity. In this case, all parameters required to establish semantic and security contexts are up to the gateway responsibility. Alternatively, the API offered by the dedicated frameworks (libraries) may be used. Using a framework, the configuration process may be reduced significantly, and the communication protocol selection has only an indirect impact on the interoperability features. In the proposed implementation, the Azure interconnection has been obtained using the above mentioned frameworks.

Azure and PubSub use different security mechanisms so in the proposed solution establishing security-context is realized independently. The *CommunicationContext* is responsible for establishing this context as an embedded negotiation phase tightly coupled with establishing interconnection.

4   Conclusion

Nowadays, the macro optimization of the industrial processes requires an integration of a vast variety of distributed applications provided by Information and Communication Technology. It requires further research on the integration of **machine - centric** Cyber-Physical Systems (CPS) with **human - centric**
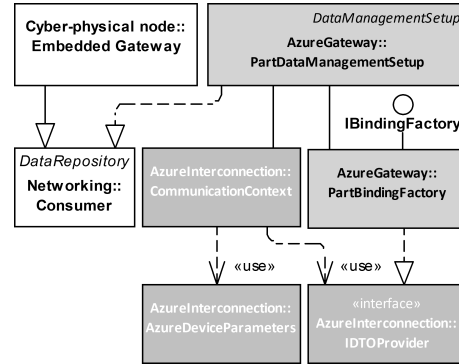
front-end in the context of new emerging disciplines, i.e. Industry 4.0 and the Industrial Internet of Things (Sect. 1).

CPS is composed using the multi-vendor components (data holding assets) interconnected atop of the Machine To Machine (M2M) communication. In many applications, the dynamic nature of the CPS must be considered. Dynamic nature means that interconnected assets may be added/removed from the network at any time. By design, CPS must typically fulfill the real-time and mobility of the assets requirements.

Highly-distributed solutions used to control/monitor a set of geographically dispersed islands of automation (e.g. virtual power plants producing renewable energy) must additionally leverage public communication infrastructure, namely the Internet. If islands of automation must be controlled over the Internet, the cloud computing concept may be recognized as a reasonable answer. Following this concept, the cloud-based supervisory control functionality is applied as a set of services employing abstraction and virtualization - two main pillars of the cloud computing paradigm. In the cloud computing concept, virtualization is recognized as the possibility of sharing the services by many users, and abstraction hides implementation details.

The main goal of this research is working out a new generic architecture and deployment scenario applicable for the integration of the **machine-centric** CPS and emerging cloud computing as a **human-centric** front-end.

If we must bother with the network traffic propagation asymmetry or mobility of the asset network attachment points the reactive relationship [25] could alleviate the challenges posed by the interactive approach. The real-time multi-vendor environment makes communication standardization especially important. To support this environment, the OPC Unified Architecture [11] interoperability standard has been selected. As it was pointed out in Sect. 2 using OPC UA Pub-Sub [2] the aggregation of nodes by the network is loosely coupled, i.e. nodes can be added and removed from the network dynamically, and nodes may represent mobile data holding assets.

From the analysis covered by Sect. 2 it is concluded that the *Embedded Gateway* archetype best suits all requirements described above. It relaxes most of the issues related to **direct interconnection** and solutions inferred from the middleware concept, i.e. **edge device** - a remote cloud agent and **field level gateway** - a dedicated custom agent. Additionally, it could be used to connect to many independent clouds at the same time. The generic domain model for the proposed interconnection archetype is presented in Fig. 1. We have highlighted that to promote the separation of concerns design principle, the gateway functionality should be implemented as a self-contained dedicated software part embedded in the core *Networking* service of the *Cyber-physical node*. The main functionality of this component is to transfer process data between *Cyber-physical network* using *Networking* services of an existing *Cyber-physical node* and *Cloud-based front-end* using interconnection services officially supported by the cloud vendor.

To promote interoperability and address the demands of the M2M communication in the context of a multi-vendor environment the prototyping should use a framework that must be compliant with the OPC UA Part 14 PubSub (Sect. 2) and support the *Reactive Interoperability* (Sect. 1) concept. We proposed to use an open-source library named *UAOOI.Networking* (*Networking* for short) (Fig. 2) for this purpose. It is worth stressing that based on this approach only dedicated functionality related to the communication with the cloud must be implemented.

We derived the final model presented in Fig. 4 by merging selected entities from the *Networking* library (Fig. 2) into the generic interconnection domain model (Fig. 1). In the proposed approach the *Embedded Gateway* is derived from the *Consumer* role implemented as a composable part aggregated by the *Reactive Application*.

The proposals are backed by proof of concept reference implementations. Prototyping addresses Microsoft Azure cloud as an example. The outcome has been just published on GitHub as the open-source (MIT licensed) repository. The proposed solutions have been harmonized with the more general concept called the Object-Oriented Internet.

The described results prove that the *Embedded Gateway* archetype implementation is possible based on the existing standalone framework supporting reactive interoperability atop the M2M communication compliant with the OPC UA PubSub standard. It is worth stressing that there is no dependency on the Client/Server session-oriented relationship. This relationship is in contrast to the architecture described in the OPC UA Part 1 [14] specification where the publisher role is tightly coupled with the **Address Space** [1] embedded component of the OPC UA Server. The real challenge of the future work is to prove that the proposed solution is flexible enough to be used as an archetype to inject the *Embedded Gateway* part into the OPC UA Client/Server to get connected with the cloud addressing the interactive relationship.

In the proposed approach the cloud interoperability is obtained by implementing a dedicated part employing out-of-band communication only without dependency on the OPC UA functionality at all. It is worth stressing that the gateway functionality is implemented as a part composable to the whole without programming skills. Because the part is composed at the runtime it makes it possible to modify its functionality later after releasing the library or even deploying the application program in the production environment.

Concluding, the paper describes a proof of concept that applying the proposed architecture and deployment scenario it is possible to integrate cloud services (e.g. **Azure IoT Central**) with the *Cyber-physical network* interconnected as one whole atop of the OPC UA PubSub. It is in contrast to interconnecting cloud-based front-end services with the *Address Space* instance exposed by a selected OPC UA server limiting the PubSub role to data exporter transferring the data out of the OPC UA ecosystem.

References

1. Opc unified architecture specification part 3: Address space model. Specification 10000-3, OPC Foundation (2017), https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-3-address-space-model/

2. Opc unified architecture specification part 14 - pubsub. Specyfication 10000-14, OPC Foundation (2018), https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-14-pubsub/

3. Ashton, K.: That 'internet of things' thing. RFID JOURNAL **2009**, 1–1 (Jun 22, 2009), https://www.rfidjournal.com/articles/pdf?4986

4. Bansal, N.: Designing Internet of Things Solutions with Microsoft Azure: A Survey of Secure and Smart Industrial Applications, chap. Microsoft Azure IoT Platform, pp. 33–48. Apress; 1st ed. edition (September 8, 2020) (09 2020)

5. Cohn, R.J., Coppen, R.J.: Mqtt version 3.1.1 plus errata 01. Tech. rep., OASIS (10 December 2015), http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

6. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/subscribe. ACM Computing Surveys **35**(2), 114–131 (2003). https://doi.org/10.1145/857076.857078

7. González, I., Calderón, A., Figueiredo, J., Sousa, J.: A literature survey on open platform communications (opc) applied to advanced industrial environments. Electronics **8**, 510 (05 2019). https://doi.org/10.3390/electronics8050510

8. Jeyaraman, R., Telfer, A.: Oasis advanced message queuing protocol (amqp) version 1.0. Tech. rep., OASIS (29 October 2012), http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html

9. Koziolek, H., Burger, A., Platenius-Mohr, M., Rückert, J., Stomberg, G.: Openpnp: A plug-and-produce architecture for the industrial internet of things. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). pp. 131–140 (2019). https://doi.org/10.1109/ICSE-SEIP.2019.00022

10. Lawton, G.: Machine-to-machine technology gears up for growth. Computer **37**(9), 12–15 (2004). https://doi.org/10.1109/MC.2004.137

11. Mahnke, W., Leitner, S.H., Damm, M.: OPC Unified Architecture. Springer, 1 edn. (May 2009)

12. Meng, Z., Wu, Z., Muvianto, C., Gray, J.: A data-oriented m2m messaging mechanism for industrial iot applications. IEEE Internet of Things Journal **4**(1), 236–246 (2017). https://doi.org/10.1109/JIOT.2016.2646375

13. Mrozek, D., Milik, M., Małysiak-Mrozek, B., Tokarz, K., Duszenko, A., Kozielski, S.: Fuzzy intelligence in monitoring older adults with wearables. In: Krzhizhanovskaya, V.V., Závodszky, G., Lees, M.H., Dongarra, J.J., Sloot, P.M.A., Brissos, S., Teixeira, J. (eds.) Computational Science – ICCS 2020. pp. 288–301. Springer International Publishing, Cham (2020)

14. Opc unified architecture specification part 1 - overview and concepts. Specyfication 10000-1, OPC Foundation (2017), https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-1-overview-and-concepts/

15. Opc unified architecture specification part 5: Information model. Specification 10000-5, OPC Foundation (2017), https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-5-information-model/

16. Pfrommer, J., Stogl, D., Aleksandrov, K., Navarro, S.E., Hein, B., Beyerer, J.: Plug and produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems. At-Automatisierungstechnik **63**(10), 790–800 (2015). https://doi.org/10.1515/auto-2014-1157

17. Postół, M.: OPC From Data Access to Unified Architecture, sec. UA Specifications, pp. 94–99. VDE VERLAG GMBH, 4th revised edition edn. (2010)
18. Postół, M.: OPC From Data Access to Unified Architecture, sec. Main Technological Features, pp. 99–104. VDE VERLAG GMBH, 4th revised edition edn. (2010)
19. Postół, M.: OPC From Data Access to Unified Architecture, sec. Information model, pp. 111–130. VDE VERLAG GMBH, 4th revised edition edn. (2010)
20. Postół, M.: Object oriented internet. In: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS). pp. 1069–1080 (2015). https://doi.org/10.15439/2015F160
21. Postół, M.: Opc ua information model deployment. Tech. rep., CAS (April 18, 2016). https://doi.org/10.5281/zenodo.2586616
22. Postół, M.: Computer Game Innovations 2018, chap. Machine to Machine Semantic-Data Based Communication: Comprehensive Survey, pp. 83–101. Lodz University of Technology Press, Łódź Poland (2018)
23. Postół, M.: Object-oriented internet. Tech. Rep. 5.1.0, GitHub (2019). https://doi.org/10.5281/zenodo.3345043
24. Postół, M.: Object oriented internet; azure gateway implementation 1.0. Tech. rep., GitHub (2020). https://doi.org/10.5281/zenodo.4361640, https://github.com/mpostol/OPC-UA-OOI
25. Postół, M.: Object-oriented internet reactive interoperability. In: Krzhizhanovskaya, V.V., Závodszky, G., Lees, M.H., Dongarra, J.J., Sloot, P.M.A., Brissos, S., Teixeira, J. (eds.) Computational Science – ICCS 2020. pp. 409–422. Springer International Publishing, Cham (2020)
26. Postół, M.: Object-oriented internet; ua part 14: Pubsub main technology features. Tech. rep., GitHub (2020). https://doi.org/10.5281/zenodo.4361640, https://commsvr.gitbook.io/ooi/reactive-communication/readme.pubsubmtf
27. Sunyaev, A.: Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies, chap. Middleware, pp. 125–154. Springer International Publishing, Cham (2020)
28. Verma, P.K., Verma, R., Prakash, A., Agrawal, A., Naik, K., Tripathi, R., Alsabaan, M., Khalifa, T., Abdelkader, T., Abogharaf, A.: Machine-to-machine (m2m) communications: A survey. Journal of Network and Computer Applications **66**, 83–105 (2016). https://doi.org/10.1016/j.jnca.2016.02.016
29. Xu, L.D., He, W., Li, S.: Internet of things in industries: A survey. IEEE Transactions on Industrial Informatics **10**(4), 2233–2243 (2014). https://doi.org/10.1109/TII.2014.2300753