Computer Standards & Interfaces xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Integrating OPC UA with web technologies to enhance interoperability

Salvatore Cavalieri*, Marco Giuseppe Salafia, Marco Stefano Scroppo

Department of Electrical Electronic and Computer Engineering (DIEEI), University of Catania, Viale A.Doria, 6, 95125 Catania Italy

ARTICLE INFO

Keywords:
OPC UA
REST architecture
Web technologies
IIoT

ABSTRACT

The so-called fourth industrial revolution features the application of modern Information & Communication Technology (ICT) concepts and technologies in industrial contexts to create more flexible and innovative products and services leading to new business models and added value. The emerging Industrial Internet of Things (IIoT) is one of the main results of this revolution. One of the most known and adopted communication protocol in the context of the fourth industrial revolution is OPC UA (IEC 62541). Although this standard already combines features coming from both industrial and ICT contexts, current literature presents several approaches aimed to introduce ICT enhancements into OPC UA in order to further improve its usability in industrial environments. Some of these approaches are based on the proposal to make OPC UA RESTful, due to many advantages of RESTful services in industrial settings. OPC UA is based on a client/server architecture and adopts an information model at the server side, whose access from the client side requires knowledge of a data model, whose structure is sometimes complex, creating some difficulties for a resource-constrained device acting as client. The paper proposes the definition of a web platform able to offer access to OPC UA servers through a REST architecture. The approach presented in the paper differs from other existing solutions, mainly because it allows to realise a lightweight OPC UA RESTful interface reducing the complexity of the basic knowledge to be held by a generic user of the platform. For this reason, the solution presented allows enhancement of OPC UA interoperability towards resource-constrained devices. The web platform has been implemented and the relevant code is available on GitHub.

1. Introduction

In the engineering and manufacturing domain, there is currently an atmosphere of departure to a new era of digitised production. Globalisation, ubiquitous presence of communication networks and the Internet, new human-machine collaboration scenarios as well as the development of complex information systems are some of the developments that are causing an industrial revolution, the fourth one.

The fourth industrial revolution has been coined with different names in the different countries; the most known is Industry 4.0. It features the application of modern Information & Communication Technology (ICT) concepts, such as Internet of Things (IoT) and Cyber-Physical Systems (CPS), in industrial contexts to create more flexible and innovative products and services leading to new business models and added value [1,2]. The emerging Industrial Internet of Things (IIoT) is one of the main results of this revolution [3].

Realisation of this novel vision may be achieved only if a big effort is really put to make interoperable the interchange of data between the different industrial applications [4]. Definition and adoption of communication standards are of paramount importance to reach this aim [5]. For this reason, during the last few years different organisations have developed reference architectures to align standards in the context of the fourth industrial revolution. Among them, there is the Reference Architecture Model for Industry 4.0 (RAMI 4.0) [6,7], which defines a reference architectural model for the current industrial revolution; it basically defines a three-dimensional matrix that can be used to position existing standards. RAMI 4.0 indicates for OPC UA [8] (one of the main communication standards currently used in industry) the role to standardise machine-to-machine communication using a uniform data format.

OPC UA allows applications to exchange information using a client/server model. OPC UA is the successor of one of the most influential technologies in the field of automation integration, the classic OPC [9].

Although OPC UA already combines features coming both from industrial and ICT contexts, several activities have been carried on during these last years aimed to introduce ICT enhancements into OPC UA in order to further improve its usability inside the fourth industrial revolution and in particular, in the IIoT.

REpresentational State Transfer or *REST* is an architectural style defined by Roy Fielding in his PhD dissertation [10]; a Web Service based

* Corresponding author.
E-mail addresses: salvatore.cavalieri@unict.it (S. Cavalieri), marcogiuseppe.salafia@unict.it (M.G. Salafia), marcostefano.scroppo@dieei.unict.it (M.S. Scroppo).

https://doi.org/10.1016/j.csi.2018.04.004
Received 17 August 2017; Received in revised form 16 April 2018; Accepted 21 April 2018

on REST is called RESTful Web Service [11]. Due to many advantages of RESTful Web Services in industrial settings as shown in [3,12], the idea of using them is frequently proposed in the context of IoT architectures [13]. For this reason, literatures presents some proposals to access OPC UA by RESTful Web Services; the REST architecture-based solutions described in [3,14,15] may be considered to this purpose.

An OPC UA Server typically offers to the OPC UA Clients a data set full of information, including variables, objects, methods, events and types; encoding rules for custom data types are also included in the set. Access to the data set requires to an OPC UA Client a fully knowledge of the OPC UA data model. Furthermore the set of services to access an OPC UA Server, may require a great amount of messages exchanged between client and server also in the case of simple requests from the client. For example, a logical connection to the server must be created by the client through several service calls also in order to realise a simple read or write access to the server [3]. Another example is when client has to retrieve a single information value of a vendor specific type, needing to access to both the value and the relevant type definition describing its structure and the encoding rules to be used.

Complex data model and intensive message exchanges may create some difficulties for a resource-constrained device acting as client. In IIoT environment, certain applications may be constrained to access to data sets of limited complexity and to limit the number of transactions needed to access to them. This mainly occur when applications run on physical devices featuring a set of limited hardware and/or software resources.

Due to the reasons just explained, the paper proposes the definition of a web platform able to offer to a generic client a simplified interface to OPC UA Servers; lightweight interface is considered both in terms of messages exchanged between client and server and in terms of basic knowledge (e.g., data model) to be held by a client. The web platform proposed in this paper is based on a REST architecture, due to the relevant advantages pointed out in [3]. The platform will be called *OPC UA Web Platform* in the remainder of the paper. The generic term *Web User* will identify a generic application which consumes the services offered by the OPC UA Web Platform; these services allow the Web User to access information maintained by OPC UA Servers, without knowledge of the OPC UA standard and without a real awareness of the presence of OPC UA Servers behind the OPC UA Web Platform, which are seen just as generic servers publishing information. Web User is required neither to be an OPC UA Client nor to implement the OPC UA communication stack (i.e., OPC UA protocol and services). Web User is only constrained to have knowledge of basic concepts detailed in the paper.

For the reasons just explained, the solution presented allows enhancement of OPC UA interoperability towards applications running on resource-constrained devices acting as Web User of the OPC UA Web Platform.

The web-based platform presented in the paper has been implemented and the relevant code is available on GitHub [16]. Other papers written by some of the authors have the same subject of this paper and they have been already published in conferences [17–19]. The web-based platform here present has been subjected to very deep enhancements during its development lasted several months. Previous papers are relevant to the early stage of this development.

The paper is organised as it follows. After an overview of the OPC UA and JSON standards used in the paper, description of the OPC UA Web Platform and its interface based on RESTful web services will be given. Case studies on simple scenarios will be presented in order to better explain the OPC UA Web Platform features. Finally, details about implementation available in [16] will be given.

2. OPC UA overview

The aim of this section is to deepen some features of the OPC UA international standard IEC 62541 needed to understand the content of the paper [8,20]. It is mainly based on a client/server communication model. Very recently, an extension of OPC UA has been defined by OPC Foundation (www.opcfoundation.org); it has been called PubSub [21] and defines a Publish/Subscribe model [22] to be added to the existing Client/Server one. At the moment, the research here presented was developed, the definition of the PubSub model was at a very early stage. For this reason, the work presented in this paper has been based only on the Client/Server model described in the following.

2.1. OPC UA AddressSpace

Inside an OPC UA Server, *OPC UA Nodes* are used to represent any kind of information, including variable instances and types. The set of OPC UA Nodes inside an OPC UA Server is called *AddressSpace* [23].

Each OPC UA Node belongs to exactly one of the following classes:

- *Variable NodeClass*, modelling values of the system. Two types of Variables are defined: *Properties* and *DataVariables*. A Property contains server-defined metadata characterising what other OPC UA Nodes represent. A DataVariable represents the data of an OPC UA Object and it may be made up by a collection of other OPC UA DataVariable Nodes. The Variable NodeClass features the *Value* attribute containing its current value; another attributed called *DataType* specifies the type for the Variable Value.
- *VariableType NodeClass*, providing type definition for Variables.
- *DataType NodeClass*, providing type definition for the Variable Value.
- *ReferenceType NodeClass*, used to define the type of a reference. In the following, description of reference will be given.
- *Method NodeClass*, modelling callable functions that initiate actions within an OPC UA Server.
- *View NodeClass*, allowing OPC UA Servers to subset the AddressSpace into Views to simplify Client access.
- *Object NodeClass*, representing real-world entities like system components, hardware and software components, or even a whole system. An OPC UA Object is a container for other OPC UA Objects, DataVariables and Methods. As the Object Node does not provide for a value, an OPC UA DataVariable Node is used to represent the data of an Object. For example, an Object modelling a file uses a DataVariable to represent the file content as array of bytes. As another example, function blocks in control systems might be represented as OPC UA Objects. The parameters of the function block (e.g., its setpoints) may be represented as OPC UA DataVariables. The Object function block might also have properties that describe its execution time and its type.
- *ObjectType NodeClass*, holding type definition for OPC UA Objects. OPC UA standard allows extending the standard ObjectTypes with additional components; the BaseObjectType is the base ObjectType and all other ObjectTypes shall either directly or indirectly inherit from it. OPC UA defines a particular ObjectType to model hierarchy among OPC UA Nodes: the FolderType. Instances of FolderType are used to organise the AddressSpace into a hierarchy of OPC UA Nodes.

The NodeClasses share some attributes, among which: *NodeId*, which identifies the OPC UA Node inside the OPC UA AddressSpace and *DisplayName*. This last attribute is used by OPC UA Clients to display the name of the OPC UA Node to the user.

OPC UA also defines *Events*, which represent specific transient occurrences; system configuration changes and system errors are examples of Events. *Event Notifications* report the occurrence of an Event. Events are not directly visible in the OPC UA AddressSpace. OPC UA Objects can be used to subscribe to Events. In particular, OPC UA defines a *Notifier* as an OPC UA Object that can be subscribed by OPC UA Clients to get Event Notifications.

OPC UA specifications define graphical symbols to represent