# Evaluation of OPC UA technology in order to minimize systems unavailability by improving M2M connectivity

João Vicente Cardoso Faria e Ivandro Cecconello

**Abstract**

Eventually the industrial automation engineer is challenged to optimize a M2M communication process. For this, it is essential to observe the infrastructure offered by the company and the resources available in legacy equipment. Developing a simple, economically viable solution that makes use of open communication standards is highly recommended. Alternatives that do not require large investments gain a significant dimension; another important factor in this decision is that the chosen solution should be able to operate harmoniously with existing legacy systems. By doing so, the company will be in a position to gradually pursue the Industry 4.0 concept and to preserve the most investments already made. This paper aims to describe OPC UA technology, test it in a controlled environment in order to meet a real production demand and justify how and why its differentiated characteristics are recommended for Industry 4.0 applications.

**Keywords**

Industry 4.0, IIoT, M2M, OPC UA, connectivity, industrial automation, PLC, *downtime*

## I. INTRODUCTION

Industry 4.0 (I4.0) is an initiative supported by the German government as a high level technology strategy in order to keep Germany in a global leading position in manufacturing [1]. I4.0 is also an oportunity to keep factories in Europe through efficient and individualized production and increasing productivity, creating investment and business opportunities and to achieve the re-industrialization targets in a sustainable way [2] as a response to the growing increase in productivity and competitiveness of emerging industrial powers [3].The German initiative was followed by several industrialized countries that also created their national programs with similar objectives. *Japan created (The Industrial Value Chain Initiative – IVI) and South Korea (Manufacturing Innovation 3.0). Countries such as the United States (Manufacturing USA), United Kingdom (Made Smarter), France (Usine du Futur), seeking through their programs to overcome the intense process of deindustrialization, reinforcing its industrial competences committed over the years and expanding the weight of production and industrial employment in the domestic economies [3]. China, with its initiative "Made in China 2025" (MIC 2025), has made extraordinary investments in the direction of becoming a highly industrialized and competitive country in every way [4]. Fig. 1 shows more than 30 countries introduced initiatives I4.0 w*ith similar *objectives*.


*Fig. 1: Industry 4.0* – global initiatives [5]

Companies challenges in the implementation of I4.0 process concept requires firstly, the identification of the maturity level and the reality of each company according to fig. 2 [5].

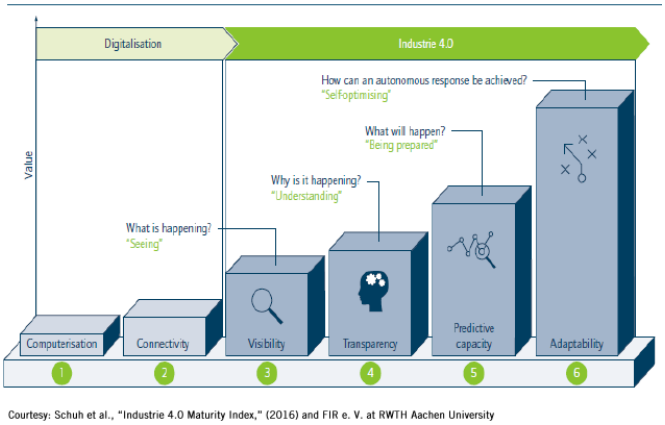


Fig. 2: I4.0 Maturity index [5]

The use of well-defined and widely accepted standards has become an urgent necessity for the successful implementation of the I4.0 concepts [6]. One of the most significant results of this was the development of the *Reference Architectural Model RAMI 4.0 and The Industry 4.0 component* [7]. They contain the fundamental aspects of I4.0 and expand the hierarchy levels of IEC 62264 (*Enterprise-control system Integration*) by adding the product level to the base and, the world connected as being the factory boundary individual at the top.

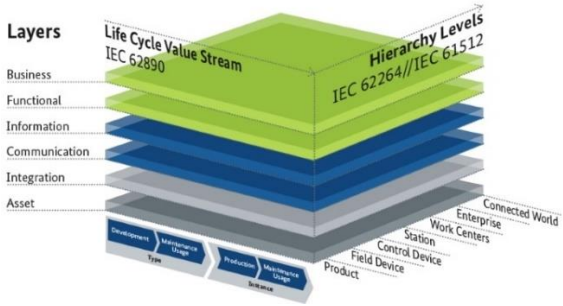


*Fig.3: The Reference Architectural Model RAMI 4.0. © Plattform Industrie 4.0* [7]

Pós-Graduação em Engenharia Industrial - Universidade de Caxias do Sul (UCS)

Emails: jvcfaria@ucs.br, iceccone@ucs.br

Data de envio: 19/02/2019
Data de aceite: 08/04/2019

http://dx.doi.org/10.18226/23185279.v7iss2p40

The horizontal axis is used to represent the life cycle of systems or products, establishing the distinction between type and instance [8]. Finally, the 6 layers define the representation of a component I4.0 in the IT structure as shown in Fig. 3 [7]. The junction of what is presented in Fig. 2 and 3 shows how essential is the concept and basis of this process the connectivity, the exchange of data and information in a standardized and safe way between devices, machines and services in different industrial segments [9]. In this sense, over time, two significant communication technologies were developed, among others: *Data Distribution Service for Real-time Systems (DDS)* and *Open Platform Communication - Unified Architecture (OPC UA)* [11] [12].

Although they have the same goal, one is the opposite of the other: DDS is data-driven (encapsulates methods and exposes data), and OPC UA is Object-oriented (encapsulates data and exposes methods) [13]; both currently have publish/subscribe capabilities. Due to the importance and relevance of these two technologies, many studies and development are being made by the OPC Foundation and Object Management Group (OMG) to use them together [14] [15].

Components I4.0 have communication capability using *Service-Oriented Architecture* (SOA) and adds semantics for virtual representation of physical objects [16]. The exchange of information, between two or more components I4.0, requires a well-defined semantics and because of this, RAMI 4.0 recommends OPC UA [7], for its ability to communicate and virtual representation of the information models [7] [17].

In the publication of "Status Report RAMI 4.0" [18] it is possible to observe that, if there is no longer the presumed definition by the technology OPC UA alleged in [7], there is a clear tendency that it will be the technology adopted or recommended by RAMI 4.0. This can be seen by the number of times that OPC UA is mentioned to the detriment of other existing technologies. This non-formal definition within the specification is also perceived in the *Industrial Internet Consortium (IIC*), equivalent to RAMI 4.0, although it does not clearly specify the recommended communication support technology, speaks repeatedly *in "data-centric publish-subscribe communications Model*" [19], DDS base [24].

As a basis for communication [6], I4.0 points to using an existing and well-established standard that is the OPC UA [20], which primarily uses a service-oriented architecture based on *client/server communication*, having received in February 2008 an extension that permit and *publish-Subscribe* messages [6] using *brokers* such as AMQP, thus ensuring interoperability between system [24].

Connectivity and OPC UA walk together in the search for I4.0, justifying the use of this technology. OPC UA can help industries integrate in the concept of I4.0 by enabling safe remote access to plant information and vertical integration [22] through *Cyber-physical Production System* (CPPS), *Computer Science* (CS) based production Technology Version, *Information and Communication Technology* (ICT) and the IEC-61499 (*Distributed Automation Systems*) [17] standard, enabling low cost to access data in industrial plants safely [23].

Given this context, the objectives of this work are to evaluate OPC UA technology, to apply it in a pilot using legacy hardware and to minimize the dependence of a centralized system (SQL database server) for the effective maintenance of M2M connectivity. The expected result is the reduction of the system's unavailability and the generation of an organizational learning in the use of this technology.

## II. THEORETICAL BACKGROUND

The OPC (*OLE for Process Control*) was introduced in 1996 as a way to protect client applications from the details of the equipment automation, supplying standardized interfaces to interact with h*ardware's c*ontrol and field devices [7]. The original specification of OPC was based on OLE Technologies (*object Linking and Embedding), COM (Component Object Model)* and *Distributed Component Object Model* (DCOM), tying it exclusively to the Microsoft platform [29]. *Microsoft, in declaring* DCOM dead at 2002, boosted the OPC Community, in addition to dealing with the problem of technological obsolescence, to meet to increasing demand for OPC support on non-Microsoft platforms [11]. The first attempt to address these problems resulted in the definition of a new standard that led to the initiative of the OPC UA 1.0 in 2009 [2]. The acronym OPC became "*Open Platform Communication*" and the new standard *OPC Unified Architecture* (OPC UA) [26].

OPC Foundation: *The interoperability Standard for Industrial automation & Other Related Domains* – is the organization dedicated to ensuring interoperability in automation by creating and maintaining open specifications that standardize the communication through the OPC UA [27]. Interoperability is understood as the characteristic of a product or system where its interfaces are fully understood, thus being able to work with other products or systems in its implementations or accesses without any restrictions [28].

### A. *Applicability*

OPC UA is applicable to all industry domains such as sensor-actuator, control systems, *Manufacturing Execution Systems* (MES), *Enterprise Resource Planning Systems* (ERP), IIoT, M2M, as well as I4.0 and MIC 2025 [4][21]. All these systems demand exchange of information [17], commands and controls of industrial processes. OPC UA defines a common infrastructure model that facilitates the exchange of information. The basic principles of the information model are:

- Use of object-oriented techniques, including hierarchy and inheritance. All instances of the same type can be treated in the same way [17]. The hierarchy type allows *Clients* working with basic types and ignoring information much specific [29].
- The type of information is exposed and can be accessed using the same mechanisms used to access instances, similar to the relational database information schema [29].
- Using different hierarchies, the same information can be exposed differently, where the information is organized according to the need to consume [29].
- The. There is no limitation on how to model the information, but the native models already serve most of the cases [17].

• Modeling is always done on the server side.

OPC UA is an independent platform technology through which various types of systems and devices can communicate. Communication occurs by sending request and response messages between *Clients and Servers*, or network messages between *Publishers* and *subscribers*. It supports secure and robust communication, ensuring the identity of the *OPC UA* applications and resisting attacks [ 21].
OPC UA was designed to provide robustness of the published data. The main feature OF all OPC *Servers* is the ability to publish event data and notifications, provide resources to detect and recover from communication failures during transfer with low *time-out* [21]. It supports a wide variety of server sizes, from a PLC to corporate servers [29], Defining different profiles identifiable by the client and servers. Interoperability is provided through three available data encodings such as *XML/text, UA Binary and JSON*, as well AS VARIOUS protocols Such as *OPC UA TCP, HTTPS and WebSockets* [21].
OPC UA applications (Fig. 4), by supporting multiple transport and coding protocols, allow the user to choose between performance and compatibility during phase of development, not limited to a previous definition of the product supplier [21].

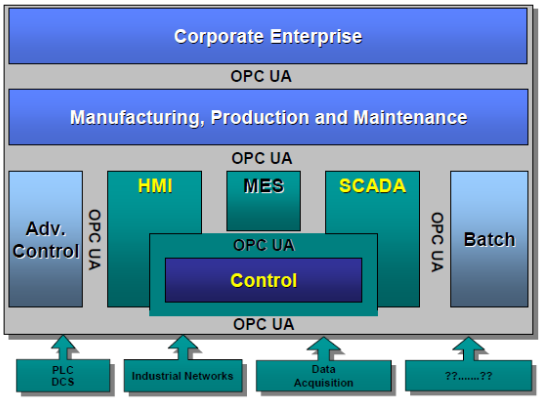


*Fig. 4: OPC UA* – Target applications [21]

## B. *Client/Server*

The *client/server interface* is defined as a set of services that enable *clients* to send requests to *servers* and receive responses from them; allows *Clients*/Subscribe to the *Servers* to receive notifications [fig. 5]. So, *servers* can automatically send the current as alarms, changes of values in data, events and results of implementation of programs [29].

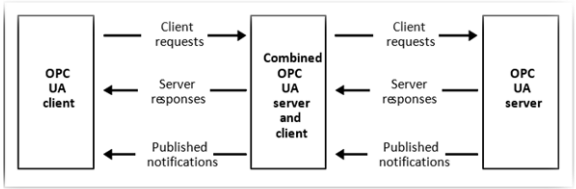


Fig. 5: Arquitetura *client/server* da OPC UA [21]

The *Subscription Service Set* is used by the *client* to create and maintain *Subscriptions,* entities that periodically publish notification messages to monitored item [21].
Once created, the existence of the *subscription* is independent of the client session created with the server, that is, a *subscription* created by a *client* can be used by another redundant *client*. For it not to be terminated, periodically the customer needs to renew interest for it [21]. The number of sequences is included in the messages allows you to identify the eventual loss of them and the request for referral by the *client*. If there is no data to be sent, the sequence number is transmitted to signal that the server is active [30].

## C. *Publish/Subscribe (PubSub)*

OPC UA defines a mechanism for *publishers* to transfer information to *subscribers* using the *pubsub* model [21].
*PubSub* is not associated with any particular messaging system; it can be mapped by several different systems, such as *User datagram Protocol (UDP),* applicable to productive environments, where demand is to regularly transmit small amounts of data to one or more devices [10].
The use of established messaging protocols, such as *Advanced Message Queuing Protocol (AMQP)* or *Message Queuing Telemetry Transport (MQTT)* with *JavaScript Object Notation (JSON) coding*, supports an integration in the cloud and enables the collection of information by modern analytical systems using flow or *batch* [21].
With *Pubsub*, OPC UA applications do not directly exchange requests and responses. Instead, *Publishers* send messages to a *message Oriented Middleware (MOM)* [31] without worrying about the existence of the *subscribers*. Similarly, *subscribers* express interest in specific types of data and process this data without worrying about the existence of *publishers*. *MOM* is an infrastructure of *Software* or *Hardware* which supports sending and receiving messages between distributed systems, depending on it as this distribution is implemented.

For scope an large number of applications, the OPC UA *pubsub* supports two variants of *MOMs*: one without a *broker*, where MOM is the network infrastructure that is capable Route messages based on *UDP multicast* and another, where *MOM* is a *broker*, making use of standard messaging protocols such as AMQP or MQTT to communicate. These messages are published to specific queues (for example, topics or nodes) that the *broker* exposes, and subscribers can listen. *Broker* can translate messages from the *publisher* 's formal protocol to the subscriber's formal messaging protocol, no matter what protocol is being used on each side [21].

## D. *Security*

Security in OPC UA cares about authenticating clients and servers, authenticating users, the integrity and confidentiality of their communications, and verifying the functionality claims. It does not Specify the circumstances in which the various security mechanisms are required [32]. This specification is crucial, but it is made by designer of a particular system and can be specified by other standards. Security measures can be configured to suit the needs of a given installation. There is a minimum set of security profiles that all OPC UA applications support, but there is no obligation to use them in all installations [21].
The application-level security depends on a secure communication channel that is active lasts the entire session and ensures the integrity of all messages that are exchanged. With this, users need to be authenticated only once during the

establishment of the application session. When a session is established, client and server applications negotiate a secure communication channel. Digital certificates (X. 509) [32] are used to identify the client and server. The server authenticates the user and authorizes subsequent requests to access objects on the server.

The security of the OPC UA presented in Fig. 6 [33] is complemented by the security infrastructure provided by most platforms that support WEB services. Transport-level security can be used to encrypt and sign messages to protect the information and integrity of messages; the algorithms used vary according to the profile chosen and with the need for the process [21].

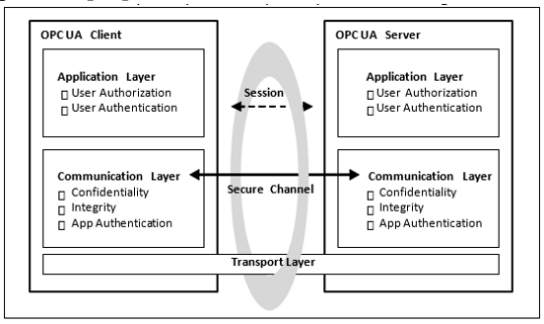


Fig. 6 – *OPC UA* security architecture [33]

In the *client/server* communication model, the *server* OPC UA (*hardware* and *software*) is the application that exposes the information and *client* is the one who requests and Works with the information [6]. The information provided by an OPC UA server is organized in the address space (*adressspace*) of the server. Services such as read and write are available with a request/response pattern used by OPC UA clients to access the information provided by an OPC UA server [29].

In this model, the client is the active entity. It chooses which server nodes (*addressspace*) and which services to use. Subscriptions are created or deleted quickly. Published only go to the client who created a subscription and the *client/server* subscription model provides reliable delivery using *buffer*, confirmations and retransmissions. This requires additional resources from the server for each connected client [11].

## III. Material and Methods

In this article is being considered only the *client/server* architecture due to its simplicity and maturity, being a *software* resource already supported by some equipment in the market, but still little applied in practice. The Architecture *Pubsub*, immensely superior in resources and performance in relation to *Client/server* traditional as described in item C above, was only officialized by The OPC Foundation in February 2018 [21], not having been identified by the author so far the application of this feature of OPC UA technology in PLCs.

### A. *Legacy system (current)*

The legacy equipment targets the analysis are *embedded PC* of the CX series of Beckhoff Automation (CX1020-0111) as illustrated by Fig. 7. This equipment, with no moving parts (fanless), combines open technology of a PC and modular I/O in DIN Rail, serial ports RS-232, ethernet ports, UPS module and K-Bus bus with various I/O. The operating system used is *Microsoft Windows Embedded CE* 6, under which it runs the PLC task supported by the Twincat 2 software – The Windows Control and Automation Technology- PLC, capable of transforming a PC into a Real-time controller, supporting all languages in IEC 61131-3 Standard [34].

Fig. 7 – PC CX1020 - Beckhoff in operation

In the application studied, the PLC program was developed in structured text; The Windows interface, with the user and corporate database, developed in C# using Microsoft Visual Studio. The communication between the Windows CE and PLC layers of the device occurs through the proprietary Beckhoff Automation Device Specification *(ADS)* protocol, allowing the reading and writing of variables under demand.

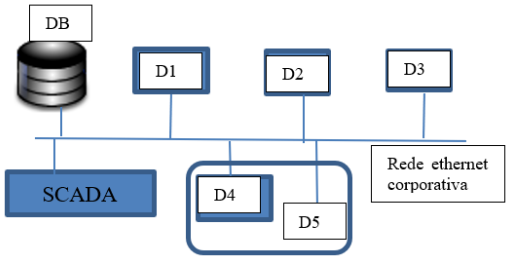


Fig. 8 – Macro view of ethernet network connectivity (author)

Fig. 8 represents the equipment and target systems of the study, where the exchange of data between them occurs basically via queries and written in a database (DB) through a program written in C# using Microsoft Visual Studio. DB represents SQL Database Server 2012, D1 to D4 Beckhoff CX1020 devices, D5 is an Allen-Bradley PLC (AB) 1768-L45S Compact GuardLogix Controller, *Supervisory Control and Data Acquisition* (SCADA) is the supervisory that program and monitor equipment, being all connected through the corporate *ethernet* network.

In the application under study D1, D2 and D3 are oriented weighing systems (OWS) that receive and send data to *DB* and guide operators rigidly throughout the production process. The equipment scheduling and distribution of activities is done through the SCADA, accessing information in the DB (production orders, *BOM*, quantity, items, routings, etc.) originated from an ERP SAP and to sequence in DB the next programs that the devices will consume. The D4 device is the standalone interface of a mixer and is responsible to update D5 with the information obtained from DB. D5 is responsible for processing raw materials (RM) weight by D1, D2 and D3 in an organized and safe way. The communication between D4

and D5 is done through the *AB EtherNet/IP* protocol. The information is shared via the database between all equipment, assuring the correct sequence of the processing of RM and the total traceability of the process. Each equipment is autonomous to fetch in DB the necessary data for its operation and also to record executed processes.
The RM correct input sequence in D4, previously weighed by D1, D2 and D3 is ensured. Each RM processed set has an identifier (ID) via barcode and/or RFID in their buckets. Before starting the weighing process the ID is read, associating the contents of the entire weighing process to it unambiguously.
The process historic (weighing) saved in the database include the ID; through this information, the D4 device can verify if that sequence set for the feeding is being followed or not, resulting in a *poka-yoke* of supply.
All communication and data exchange between these equipment, except for that occurring between D4 and D5, is through the database. M2M communication is supported by the corporative ethernet network infrastructure and remote centralized SQL database. The possible communication failures in this environment are mainly associated with equipment D1, D2, D3 and D4, the corporative ethernet network and the database. The greatest risk to communication in this process lies in the eventual unavailability of DB, resulting from the server dropping or simply failure of the corporative communication link. In this eventual failure, the whole process of M2M communication is compromised and the equipment goes to work without exchanging data between them, making use only of the information acquired before the connection loss with DB, until they are consumed and to demand updating.

### B. *Chosen solution*

This work proposes to implement an additional communication resource using OPC UA technology between D4 (information consumer) and devices D1, D2 and D3 (information producers). Thus, in the eventual temporary unavailability of DB caused by various reasons (remote network infrastructure failure, disk space missing, inappropriate maintenance, etc.), considering that the local ethernet network infrastructure remains (switches and network cabling), the new feature will ensure continuity of information exchange between the equipment until the completion of the production order in progress. After recovering access to DB, the historic stored by the devices will be saved and the production downtime will have been minimized.

### C. *Implementation*

The availability of the OPC UA feature on the Beckhoff equipment using TwinCAT 2 (TC2) on the Windows CE platform is obtained from the installation of TS6100-0030-Twincat OPC-UA Server CE Communication Library [37].The *Server* enables *Clients* OPC UA to access *NameSpace*; features are also available based on function blocks in the Standard PLCopen that allow these equipment to communicate with others OPC UA servers [37].

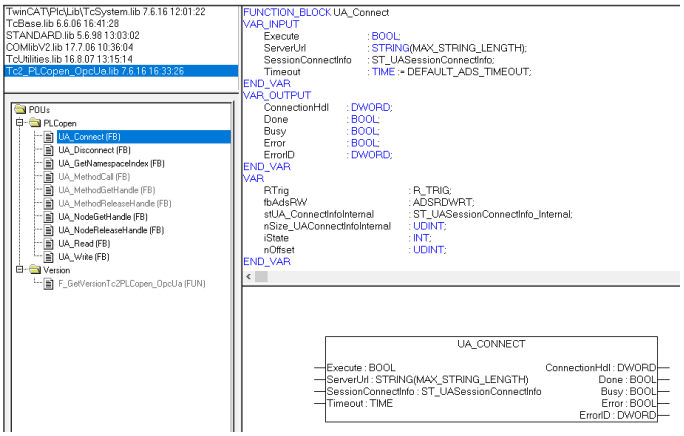

Fig. 9 – PLCopen standard functions of TS6100-0030 [35]

These two sets of functionalities form the basis of the proposed system, as shown in Fig.10 of the [35].

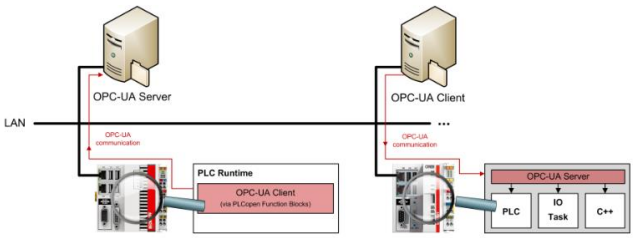

Fig. 10 – Connectivity via OPC UA [35]

In order to these variables used in a program to become accessible, it is necessary to configure the *namespace* on the OPC UA server. The *namespac*e describes the mapping structure of the variables within the PLC program, enabled in the TC2 according to the following illustrative examples:
bVariable1 : BOOL; (*~ (OPC:1:some description) *)
bVariable2 : BOOL;

In the example above, the control "bVariable1" is declared to be of the boolean type and the visibility via OPC UA is ensured through the pseudo comment "(* ~ (OPC: 1: some comment) *)", signaling that the bVariable1 is accessible by OPC UA; the variable "bVariable2 ", of the same type, will not be available at OPC UA Server. TwinCAT 3 (TC3) syntax differs from that shown above [35].

Fig. 11 represents the interaction that will occur between the existing OPC UA *servers* and *clients* in the equipment.

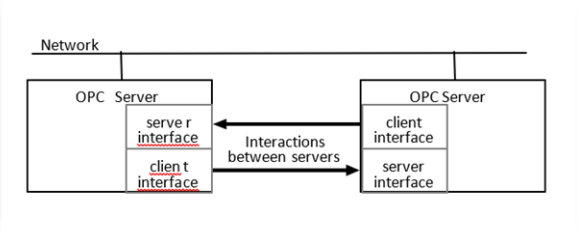

Fig. 11 – Peer-to-peer interactions between servers [29]

The function blocks [Fig. 9] provided by TS6100-0030 [38] exist to support the connection to the OPC UA server, read and write according to Beckhoff's documentation for the TC2 [37], but these functionalities could not be confirmed by the author in this study. On testing the most basic function like

connection between *client* and *server* resulted in connection failure (*ADS error 0x6-target port not found*) [37].
In contact with the manufacturer, it was recommended to use the latest TC3 replacing TC2 in order to have desired resources available,
To overcome the non-availability of equipment with TC3 and to check the above information, we used the feature of transforming a conventional PC, running Windows 10, into an equivalent PLC through the installation of TC3. With software development and execution of PLC tasks, the equipment equivalent to D4 was simulated with TC3.

### D. *Software Tools used*

In the configuration and programming of the devices with TC2 (D1, D2 and D3), the software tools used was the TwinCAT v 2.11.2301, consisting of the *PLC Control* version v 2.11.0 (Build 2618) and the *System Manager* version v 2.11.0 (Build 2285) by Beckhoff Automation
In the configuration and programming of the device with TC3 (D4), the software tool used was the TwinCAT *version* v 3.1.0.4422, which includes system Manager version v 3.1.0 (Build 4210) with "The Twincat Engineering system and Runtime System" from Beckhoff Automation, integrated into the Microsoft Visual Studio Community 2017 version 15.7.2 Microsoft Development environment.

For preliminary testing of access and parallel monitoring of the available data and performance of OPC servers in devices D1, D2 and D3, it was used the Prosys OPC UA Client v 3.1.4-293 software of Prosys OPC Ltd., fig. 12.

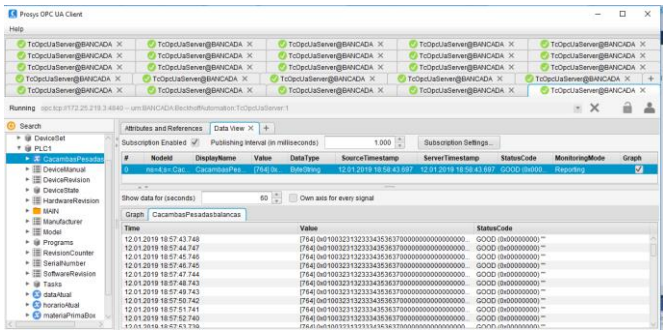

Fig. 12 – Software Prosys OPC UA Client

### E. *Data structure*

To achieve the objective of transferring to the D4 device, the production flow controller, the information of the weight buckets codes and other relevant information from devices D1 to D3, it was necessary to create data structures that organize them logically and transparently.
On the OPC UA server side, devices D1 to D3, appropriate data structures were created to organize the information to be available [fig. 13].

```
TYPE TRegistroCacambasCodigos :
STRUCT
    Balanca: WORD;
    OP: STRING;
    OP_MAE: STRING;
    DadosCacambas: ARRAY[0..99] OF TipoCacambaBalanca;
END_STRUCT
END_TYPE
```

```
TYPE TipoCacambaBalanca :
STRUCT
    Carga:WORD;
    Cacamba:WORD;
    CodigoCacamba:WORD;
END_STRUCT
END_TYPE
```

Fig. 13 – Data types created on the *servers*.
The data created based on Fig. 13 becomes visible to OPC UA clients through the following syntax in the declaration of the objects in Fig. 14.

```
VAR_GLOBAL PERSISTENT

HISTORICO : ARRAY[0..1000] OF TRegHistorico;

CacambasPesadasbalancas: TRegistroCacambasCodigos;   (*'' (OPC : 1 : enabled for OPC) *)
```

Fig. 14 – Objects declared based on types in Fig. 13.

On the *client* side so that it was possible to read/write, it WAS created in TC3 the data types and structures of Fig. 15.

```
TYPE TipoCacambaBalanca :
STRUCT
    Carga:WORD;
    Cacamba:WORD;
    CodigoCacamba:WORD;
END_STRUCT
END_TYPE
```

```
TYPE TRegistroCacambasCodigos :
STRUCT
    Balanca: WORD;
    OP: STRING;
    OP_MAE: STRING;
    //DadosCacambas: ARRAY[0..499] OF TipoCacambaBalanca;
    DadosCacambas: ARRAY[0..99] OF TipoCacambaBalanca;
END_STRUCT
END_TYPE
```

Fig. 15 – Data types and structures in TC3.


```
TYPE TipoCacambaBalanca :
STRUCT
    Carga:WORD;
    Cacamba:WORD;
    CodigoCacamba:WORD;
END_STRUCT
END_TYPE
```

```
TYPE TRegistroCacambasCodigos :
STRUCT
    Balanca: WORD;
    OP: STRING;
    OP_MAE: STRING;
    //DadosCacambas: ARRAY[0..499] OF TipoCacambaBalanca;
    DadosCacambas: ARRAY[0..99] OF TipoCacambaBalanca;
END_STRUCT
END_TYPE
```

Fig. 15 – Data types and structures in TC3.


The data structures (Fig. 15) created will receive the information from the visible data of the OPC UA servers. For testing purposes, the following objects marked in yellow were created:

```
VAR_GLOBAL
    (* These global connection and namespace settings need to be adapted to the runti
    sOpcUaServerUrl : STRING(MAX_STRING_LENGTH) := 'opc.tcp://localhost:4840'; (*'ops
    sNamespaceUri : STRING(MAX_STRING_LENGTH) := 'urn:BeckhoffAutomation:Ua:PLC1'; //

    CacambasPesadasBalancasGeral: ARRAY[1..20] OF TRegistroCacambasCodigos;

    CacambasPesadasBalancasGeral2: ARRAY[1..20] OF TRegistroCacambasCodigos;

    sOpcUaServerUrlArray : ARRAY[1..20] OF STRING(MAX_STRING_LENGTH);
    sNamespaceUriArray   : ARRAY[1..20] OF STRING(MAX_STRING_LENGTH);
    bHabilitaServer      : ARRAY [1..20] OF BOOL ;
    iIndiceBalanca : INT :=1;
    iIndiceBalanca1 : INT :=1;
    iIndiceBalanca2 : INT :=1;
    iIndiceBalanca3 : INT :=1;
    iIndiceBalanca4 : INT :=1;


END_VAR
```

Fig. 16 – Data structure that will receive the information

The two sets of *arrays* with twenty positions each, fig. 16, can receive information of up to 40 different equipment. The dimensioning, far beyond the actual need for production, aimed at enabling tests simulating the existence of a high number of *servers* and thus evaluating eventual LIMITS OF communication via OPC UA.

### *F. Software development*

On the server side, it is only necessary that the information be made available in an organized manner as the data is generated. It will not be the object of this article to demonstrate how this occurs, nor how the data obtained via OPC UA will be consumed by the *client device.* The goal is to demonstrate how the *Client* search for information in *Server* from the program developed using THE TC3 tool within the proposed architecture.
Fig. 16 shows the main program developed. It is possible to see the attribution of values to the structures required for OPC UA communication and the call of other programs that will execute the necessary processes for this communication to occur.

```
PROGRAM MAIN
VAR
    fbGetHostname : FB_GetHostName;
    {attribute 'OPC.UA.DA':='1'}
    nCounter2 : INT;
    {attribute 'OPC.UA.DA':='1'}
END_VAR
```

```
nCounter2:= nCounter2+1;
IF (nCounter2>2000) THEN
    nCounter2:=-2000;
END_IF

fbGetHostname(bExecute := TRUE);
IF NOT fbGetHostname.bBusy THEN
    IF NOT fbGetHostname.bError THEN
        sNamespaceUri := 'urn:BeckhoffAutomation:Ua:PLC1';

        sOpcUaServerUrlArray[1]:='opc.tcp://172.25.219.3:4840';
        bHabilitaServer[1]:=TRUE;
        sOpcUaServerUrlArray[2]:='opc.tcp://172.25.220.231:4840';
        bHabilitaServer[2]:=TRUE;
        //repete as estruturas para os demais índices
        sOpcUaServerUrlArray[3]:='opc.tcp://172.25.219.3:4840';
        bHabilitaServer[3]:=TRUE;
        sOpcUaServerUrlArray[4]:='opc.tcp://172.25.220.231:4840';
        bHabilitaServer[4]:=TRUE;
        sOpcUaServerUrlArray[5]:='opc.tcp://172.25.219.3:4840';
        bHabilitaServer[5]:=TRUE;
        sOpcUaServerUrlArray[6]:='opc.tcp://172.25.220.231:4840';
        bHabilitaServer[6]:=TRUE;
        sOpcUaServerUrlArray[7]:='opc.tcp://172.25.219.3:4840';
        bHabilitaServer[7]:=TRUE;
        sOpcUaServerUrlArray[8]:='opc.tcp://172.25.220.231:4840';
        bHabilitaServer[8]:=TRUE;
        sOpcUaServerUrlArray[9]:='opc.tcp://172.25.219.3:4840';
        bHabilitaServer[9]:=TRUE;
        sOpcUaServerUrlArray[10]:='opc.tcp://172.25.220.231:4840';
        bHabilitaServer[10]:=TRUE;
        sOpcUaServerUrlArray[11]:='';
        bHabilitaServer[11]:=FALSE;//limita conexões em 10

        sNamespaceUriArray[1]:='urn:BANCADA:BeckhoffAutomation:Ua:PLC1';
        sNamespaceUriArray[2]:='urn:SL-BALANCA-07-I:BeckhoffAutomation:Ua:PLC1';
        sNamespaceUriArray[3]:='urn:BANCADA:BeckhoffAutomation:Ua:PLC1';
        sNamespaceUriArray[4]:='urn:SL-BALANCA-07-I:BeckhoffAutomation:Ua:PLC1';
        sNamespaceUriArray[5]:='urn:BANCADA:BeckhoffAutomation:Ua:PLC1';
        sNamespaceUriArray[6]:='urn:SL-BALANCA-07-I:BeckhoffAutomation:Ua:PLC1';
```

The main program is the "*Main*" and will call the others, among them:
PRG_READ_TC2_STRUCT_DIVERSOS();
PRG_READ_TC2_STRUCT_DIVERSOS_2();
PRG_READ_TC2_STRUCT_DIVERSOS_3();
PRG_READ_TC2_STRUCT_DIVERSOS_4();

The *MAIN* Program data declaration is also used to exemplify the definition of object accessible via OPC UA on TC3, whose syntax is distinct from THE previously presented TC2.

The programs PRG_READ_TC2_STRUCT_DIVERSOS () listed above and the others below it, aim to fetch the data in the structures created responsible for storing the information. They all have the same basic structure and they were used to test the behavior of OPC UA technology in multiple connections, as well as the speed of communication.

All tests were performed using the transport via TCP and with the security options of the session and the disabled messages.

Fig. 17 shows the partial variable declarations of the PRG_READ_TC2_STRUCT_DIVERSOS () program. The same declaration structure is followed by the other programs executed by the main program *Main*.

```
        sNamespaceUriArray[7]:='urn:BANCADA:BeckhoffAutomation:Ua:PLC1';
        sNamespaceUriArray[8]:='urn:SL-BALANCA-07-I:BeckhoffAutomation:Ua:PLC1';
        sNamespaceUriArray[9]:='urn:BANCADA:BeckhoffAutomation:Ua:PLC1';
        sNamespaceUriArray[10]:='urn:SL-BALANCA-07-I:BeckhoffAutomation:Ua:PLC1';
        sNamespaceUriArray[11]:='';
        //Chama os diversos programas, cada um com 10 conexões (limite máximo por programa
        PRG_READ_TC2_STRUCT_DIVERSOS();
        PRG_READ_TC2_STRUCT_DIVERSOS_2();
        PRG_READ_TC2_STRUCT_DIVERSOS_3();
        PRG_READ_TC2_STRUCT_DIVERSOS_4();

    END_IF
END_IF
```

Fig. 16 – Program *Main* in TC3

The main program is the "main" and it will call the others such as:
PRG_READ_TC2_STRUCT_DIVERSOS();
PRG_READ_TC2_STRUCT_DIVERSOS_2();
PRG_READ_TC2_STRUCT_DIVERSOS_3();
PRG_READ_TC2_STRUCT_DIVERSOS_4();

The *MAIN* Program Data declaration is also used to exemplify the definition of objects accessible by OPC UA on TC3, whose syntax is distinct from the previously presented TC2.

The Program PRG_READ_TC2_STRUCT_DIVERSOS () listed above and the others below it, intend to fetch data in the structures created responsible for storing the information. They all have the same basic structure and they were used to test the behavior of OPC UA technology in multiple connections, as well as the communication speed.

All tests were performed using the transport via TCP and with the security options of the session and the disabled messages.

Fig. 17 shows the partial variable declarations of the PRG_READ_TC2_STRUCT_DIVERSOS () program. The same declaration structure is followed by the other programs executed by the main program "*Main*".

```
PROGRAM PRG_READ_TC2_STRUCT_DIVERSOS
VAR
    SessionConnectInfoArray     : ARRAY[1..10] OF ST_UASessionConnectInfo;
    nConnectionHdlArray         : ARRAY[1..10] OF DWORD;

    (* Declarations for UA_GetNamespaceIndex *)
    fbUA_GetNamespaceIndexArray : ARRAY[1..10] OF UA_GetNamespaceIndex;
    nNamespaceIndexArray        : ARRAY[1..10] OF UINT;

    (* Declarations for UA_NodeGetHandle *)
    fbUA_NodeGetHandleArray     : ARRAY[1..10] OF UA_NodeGetHandle;
    NodeIDArray                 : ARRAY[1..10] OF ST_UANodeID;
    nNodeHdlArray               : ARRAY[1..10] OF DWORD;

    (* Declarations for UA_Read *)
    fbUA_ReadArray              : ARRAY[1..10] OF UA_Read;
    stIndexRangeArray           : ARRAY[1..10] OF ARRAY [1..nMaxIndexRange] OF ST_UAIndexRange;
    nIndexRangeCountArray       : ARRAY[1..10] OF UINT;
    stNodeAddInfoArray          : ARRAY[1..10] OF ST_UANodeAdditionalInfo;
```

Fig. 17 – Software developed for testing of OPC UA features in TC3 (complete content in Appendix 1)

For a better understanding of the process, the flowchart shown in Fig. 18 represents, in a simplified way, the flow of communication control involving 10 objects in each program. As you can see, that objects once created, are saved and reused in the next communication if no error is detected. Thus, the longest time demanded by the communication is associated with the creation of objects is avoided and the process is optimized. In the event of an error, the program recreates the objects and saves them again for the next use.

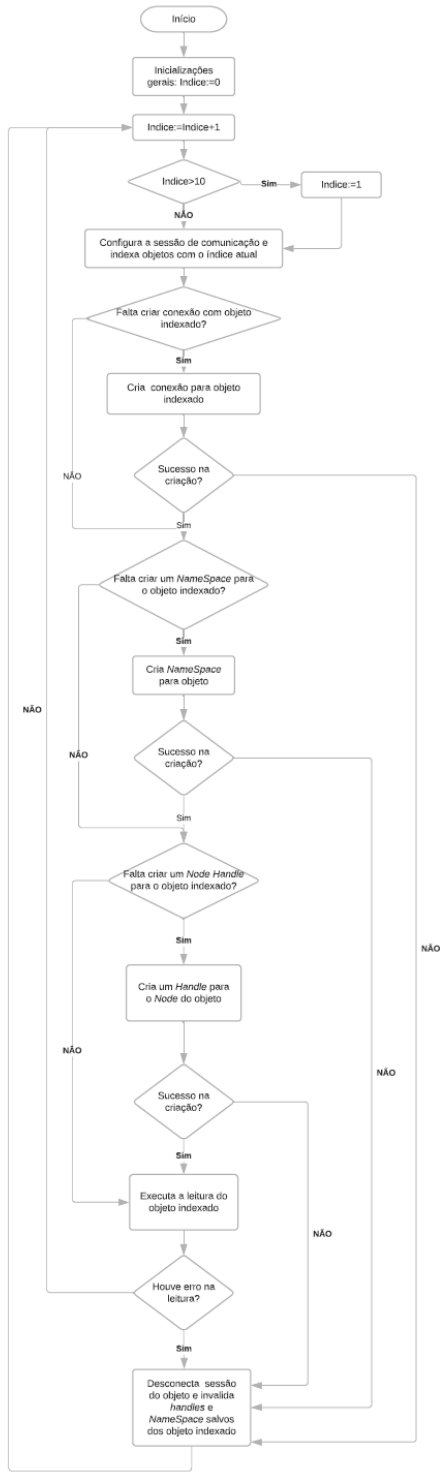


Fig. 18 – Simplified flowchart of the program PRG_READ_TC2_STRUCT_DIVERSOS

## IV. RESULTS

Prosys OPC UA Client software was used in preliminary tests to access existing data on the servers has proved to be very efficient and easy to use when compared to other tools available on the market. The existing features in Prosys have fully met the needs for validation of the proposal.

The initial tests intend to confirm the functionality of the features provided by the OPC UA *server* on the evaluated devices; it included data reading and access to structured information (fig. 12), as defined by the OPC Foundation. The *subscription* feature in the Prosys software was tested with a minimum allowable interval of 1 sec, and the object reads with 30,000 bytes in size. Satisfactory results remained after being quadrupled connections in the client. The number of simultaneous accesses (multiple connections), demonstrated communication efficiency with the same server even using Internet and depending on a network Wi-Fi with a speed of 15 Mbits.

After validated device server resources [fig. 12], the next step was to test client/server communication between Beckhoff equipment, starting with a simple reading and writing of a integer type variable until reading of complex structures involving a large amount of data, culminating with the software structure presented in fig. 17 developed on the *client* side TC3.

Performed tests showed despite of the speed in get a large amount of data, the time consumed to read a simple variable was practically the same, with no proportionality in relation to the size of the data block read. The hypothesis taken into consideration was the longest time spent during communication was the connection process, i.e., creation of the session, get *nameSpace* index and object access *handle creation*, limiting the amount of possible communications to less than 4 readings per second, independent if an object had size of two or thousands bytes.

In order to prove above hypothesis, the software structure presented in fig. 18 was developed, where it was eliminated from the time consumed in each communication the parcel required for creation of *sessions*, *nameSpace index* and *handles* access to objects. In it, once the resources (*sessions*, *nameSpace index* and *handles*) are created and allocated in appropriate data structures, they are reused in the next communication. Case we have any errors in that process, the standard communication procedure is resumed, impacting again on the communication performance.

Using that strategy represented by fig. 17, it was eliminated the major cause of delay observed in the *client/server* communication, enabling connection and reading of up to 10 concurrent objects.

The limit of 10 simultaneous active connections in a single program structure like fig. 17 was identified using the trial and error method. The limitation, verified in the tests with *client* of TC3, conflicted with the results presented on fig. 12, where there were 25 active connections running without problems, indicating that realized limitation was not in the *server* but somehow in the TC3 *client*. This limitation was overcome, from a practical standpoint by creating multiple program units, each with the maximum number of connections allowed. Because of this, it was concluded that the limitation is associated with unit of program and not with OPC UA *client* itself.

In the application developed, 4 units of programs were used, resulting in 40 independent connections and getting 764 bytes per second each. The information read was stored in different memory structures (fig. 17), permitting full monitoring of the process. The maximum number of concurrent connections possible in an application was not identified using the features showed. The investigation of these limits may be a subject to future studies case be necessary to extrapolate the number of connections tested here.
All tests were performed without using any additional security features in the communication process, not being it demanded by the actual application. However, being a security one of the most emphasized points in the OPC UA technology, we sought to verify the behavior of the TC2 servers in this item, using Prosys software [Fig. 12] to test the availability [Fig. 19].

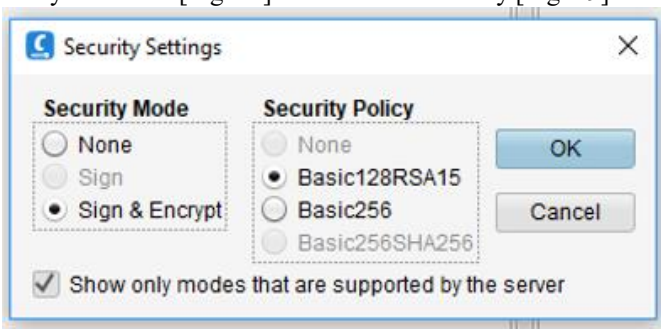


Fig. 19 – OPC UA connection security modes setting via Prosys.

The "Sign & Encrypt" security modes with "*Security Policy*" *Basic128RSA15* and *Basic256* are supported by server devices as tests performed using *anonymous user*. You must follow the procedures described in [37] in order to transfer certificates between *client* and *server.*
The proposed architecture, after replacing D4 device (fig. 8), originally a CX1020, by equipment type CX2020 with TC3 (CX2020-0115) [36] and communication library TF6100-0030 [37] resulted on become available planned communication resources. Replacing the original equipment D4 with TC2 for a new one with TC3 will require investments; the other ones with TC2(D1, D2 and D3) were validated and meet planned needs. In this new scenario, although we have no equipment zero cost, we will only need to update 25% of them.

Another important result of this study was to detect there are eventual omissions in manufactures documentation. Therefore, the strategy of developing a pilot project is highly recommended. Through the pilot, it will be possible to confirm or not the effectiveness of cited resources in the documentation and thus be more assertive in the investment needs.

## V. Conclusions:

M2M connectivity improvement was achieved after a thorough study of the resources existing in legacy equipment, as part of a well-defined strategy, organized and aligned with the concept of I4.0 to reach the desired goals. This meets what it was asserted by [1], where companies, before making any investments, must establish appropriate specifications to standardize the factory and thus gradually construct an environment I4.0.
Efforts to implement actions aligned with the concept of I4.0 in relation to integration are confronted with concepts that promise high efficiency and flexibility, but do not advance in concrete recommendations for actions [39]. The guide presented in [40] describes a structured way how to proceed in this search, but is mainly the conceptual part, without entering concrete *cases* of application. Nevertheless, [40] can be considered a significant evolution in this sense by proposing tools that will assist in this process.
OPC UA technology choice ensures interoperability between systems, one of the biggest efforts of I4.0 according to [24]. Tests proved the real possibility of the technology application on legacy equipment, resulting in improvements in M2M communication. Additionally, using OPC UA in them allows externalize information in an organized, secure and modern way, without necessarily demanding high investments.

A major advantage of the proposal is the decentralization of communication. In the model studied the equipment have their own OPC UA *server*, which enables *client's* direct access to them without dependence on a standard centralized server. So, the architecture of fig. 20 allows an optimization of the use of equipment, network and results in a better response time during communication.

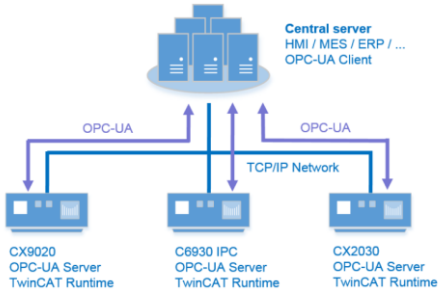


Fig. 20 – Decentralized model of OPC UA servers [38].

Using a conventional architecture with centralized, though functional, OPC UA server would result in minimal gains. The same fault modes perceived in relation to the DB server [Fig. 8] would be transferred to the remote OPC UA server, that is, a possible failure of that server would compromise M2M communication in the same way.

Tests carried out confirmed the feasibility of applying the OPC UA technology to improve M2M connectivity in what was proposed, demonstrating how some legacy systems can be levered without involving large investments. The performance of the system and the results that can be achieved will depend on the application developer's ability to make the best use of the resources.

## VI. Thanks

Authors thank to Fras-le company for the available resources and for encouraging the continuous processes improvement, as well as Beckhoff Automation by the technical support and doubts clarification.

## VII. BIBLIOGRAPHY


[1] Jian-Yu Chen, Kuo-Cheng Tai and Guo-Chin Chen. "Application of Programmable Logic Controller to Build-up an Intelligent Industry 4.0 Platform", IFAC-PapersOnLine 49-25 (2016) 008–012, 2016, Available at https://www.sciencedirect.com/science/article/pii/S2212827117302627

[2] Dr. Jörg Mutschler. "Industrie 4.0 – From Vision To Reality (Implementation among German Companies)", VDMA, 2016, available at https://docplayer.net/28404610-Industrie-4-0-from-vision-to-reality.html, Web. 27 Oct. 2018

[3] Dra. Mª Cristina Penido de Freitas (IEDI), "National strategies for Industry 4.0", 2018. Available at: https://iedi.org.br/media/site/artigos/20180705-estrategias_nacionais_para_a_industria_4_0.pdf, Web. 28 Oct. 2018

[4] Institute for Security & Development Policy. "Made in China 2025", BACKGROUNDER - June 2018, Available at: http://isdp.eu/content/uploads/2018/06/Made-in-China-Backgrounder.pdf, Web. 27 Oct. 2018

[5] Stephen Ezell. "How countries are supporting Manufacturing Digitalization", ITIF, 2018, Available at: http://www2.itif.org/2018-gtipa-summit-stephen-ezell.pdf, Web. 28 Oct. 2018

[6] Peter Adolphs, Heinz Bedenbender, Dagmar Dirzus et al., "Status Report - Reference Architecture Model Industrie 4.0 (RAMI4.0)," VDI and ZVEI, 2015. Available at: https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2016/januar/GMA_Status_Report__Reference_Archtitecture_Model_Industrie_4.0__RAMI_4.0_/GMA-Status-Report-RAMI-40-July-2015.pdf , Web. 18 Oct. 2018

[7] Dr. Karsten Schweichhart. "Reference Architectural Model Industrie 4.0 (RAMI 4.0) - An Introduction", 2015, Available at: https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf, Web. 24 Sept. 2018

[8] F. Zezulka, P. Marcon, I. Vesely and O. Sajdl. "Industry 4.0 – An Introduction in the phenomenon", ScienceDirect, IFAC-PapersOnLine 49-25 (2016) 008–012, Available at: https://ac.els-cdn.com/S2405896316326386/1-s2.0-S2405896316326386-main.pdf?_tid=82478c40-204c-4dd1-aac3-f5e30aa862bb&acdnat=1548889712_f770856f26896a55638c86aaf25682dc , Web. 24 Sept. 2018

[9] Stefan Hoppe. "There Is No Industrie 4.0 without OPC UA", 2017. Available at: https://opcconnect.opcfoundation.org/2017/06/there-is-no-industrie-4-0-without-opc-ua/, Web. 09 Oct. 2018

[10] OPC Foundation. "OPC Unified Architecture Specification, Part 14: PubSub Release 1.04", 2018. Available at: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-14-pubsub/, Web. 26 May 2018

[11] Angelo Corsaro. "A Tale of Two Industrial IoT Standards: DDS and OPC-UA",2016. , Available at: https://www.rtinsights.com/dds-opc-ua-industrial-iot-standards/, Web. 14 Oct. 2018.

[12] Shiyong Wang, Jian Ouyang, Di Li and Chengliang Liu, "An Integrated Industrial Ethernet Solution for the Implementation of Smart Factory",2017 Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8096995/, Web. 29 Apr. 2018

[13] Stan Schneider, PhD ( RTI CEO, IIC Steering Committee) and Thomas Burke (President, The OPC Foundation). "The Inside Story: How OPC UA and DDS Can Work Together in Industrial Systems", 2017, Available at: https://pt.slideshare.net/RealTimeInnovations/the-inside-story-how-opc-ua-and-dds-can-work-together-in-industrial-systems, Web. 14 May 2018

[14] OPC Foundation and Object Management Group (OMG), "Position Paper describing how they will cooperate with their communication technologies, OPC Unified Architecture (OPC UA) and Data Distribution Services (DDS) ", 2017, Available at: https://opcfoundation.org/wp-content/uploads/2016/04/OPCF-OPCUA-OMG-DDS-Positionspaper-final-v1.pdf, Web. 14 Oct. 2018

[15] Bryon Moyer. "Collaboration Between OPC UA and DDS How Are These Protocols Different? How Are They Similar?", Electronic Engineering Journal, 2016. Available at: http://www.eejournal.com/article/20160516-opc/, Web. 14 Oct. 2018

[16] Behrad Bagheri, NSF I/UCRC for Intelligent Maintenance Systems (IMS) and Jay Lee (University of Cincinnati). "Big future for cyber-physical manufacturing systems", Editor Design World | September 23, 2015, Available at: https://www.designworldonline.com/big-future-for-cyber-physical-manufacturing-systems/, Web. 15 Oct. 2018

[17] Florian Paukera, Thomas Fruhwirth, Burkhard Kittla and Wolfgang Kastnerb. "A systematic approach to OPC UA information model design",2016, Procedia CIRP 57 ( 2016 ) 321 – 326, Available at: https://www.sciencedirect.com/science/article/pii/S2212827116312100, Web. 29 Apr. 2018

[18] Dr. Peter Adolphs, Dr. Heinz Bedenbender, Dr. Dagmar Dirzus, Martin Ehlich, Prof. Ulrich Epple, Martin Hankel, Roland Heidel, Dr. Michael Hoffmeister, Haimo Huhle et al. "Reference Architecture Model Industrie 4.0 (RAMI4.0)", 2015, Available at: https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2016/januar/GMA_Status_Report__Reference_Archtitecture_Model_Industrie_4.0__RAMI_4.0_/GMA-Status-Report-RAMI-40-July-2015.pdf,Web. 18 Oct. 2018

[19] Industrial Internet Consortium (IIC), "The Industrial Internet of Things Volume G1: Reference Architecture", IIC:PUB:G1:V1.80:20170131, 2017, Available at: https://www.iiconsortium.org/IIC_PUB_G1_V1.80_2017-01-31.pdf,Web. 11 Nov. 2018

[20] Jakob Axelsson, Joakim Fröberg and Peter Eriksson. "Towards a System-of-Systems for Improved Road Construction Efficiency Using Lean and Industry 4.0",2017, Available at: https://www.researchgate.net/publication/326949888_Towards_a_System-of-Systems_for_Improved_Road_Construction_Efficiency_Using_Lean_and_Industry_40, Web. 03 Oct. 2018

[21] OPC Foundation. "OPC Unified Architecture Specification, Part 1: Overview and Concepts Release 1.04", 2017. Available at: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-1-overview-and-concepts/,Web. 03 May 2018

[22] Marcelo V. García, Edurne Irisarri, Federico Pérez, Elisabet Estévez, and Marga Marcos. "An Open CPPS Automation Architecture based on IEC-61499 over OPC-UA for flexible manufacturing in Oil&Gas Industry", IFAC PapersOnLine 50-1 (2017) 1231–1238 Available at: https://www.sciencedirect.com/science/article/pii/S2405896317306791, Web. 29 Apr. 2018

[23] Dr. Oscar Carlsson. "Engineering of IoT Automation Systems", EISLAB Lule˚a University of Technology, ISBN 978-91-7583-792-50 Platform", 2017, Available at http://www.diva-portal.org/smash/get/diva2:1060783/FULLTEXT01.pdf, Web. 27 Oct. 2018

[24] Thomas Bangemann, Magdeburg Christian Bauer, Karlsruhe Heinz Bedenbender, Düsseldorf Annerose Braune, Dresden Christian Diedrich et al, "Industrie 4.0 Service Architecture - Basic concepts for interoperability", VDI/VDE Society Measurement and Automatic Control (GMA), 2017, Available at: https://docplayer.net/58377070-Status-report-industrie-4-0-service-architecture-basic-concepts-for-interoperability.html,Web. 15 Oct. 2018

[25] Jürgen Lange, Frank Iwanitz and Thomas Burke. "OPC from Data Access to Unified Architecture." 4th rev. Edition. Berlin: VDE VERLAG GMBH, 2010. ISBN 3-978-3-8007-3242-5

[26] OPC Foundation. "What is OPC?", Available at: https://opcfoundation.org/about/what-is-opc/,Web. 20 Sep. 2018

[27] OPC Foundation. "What is the OPC Foundation?", Available at: http://www.opcfoundation.org/Default.aspx/01_about/01_history.asp?MID=AboutOPC,Web. 15 Oct. 2018

[28] John de Wardt. "Interoperability in Drilling Systems Automation: not just about data transfer", DE WARDT AND COMPANY INC, 2017, Available at: https://dewardt.com/wp-content/uploads/2017/11/2-White-Paper-Nov-2017.pdf,Web. 11 Nov. 2018

[29] Wolfgang Mahnke, Stefan-Helmut Leitner and Matthias Damm. "OPC Unified Architecture", e-ISBN 978-3-540-68899-0, 2009

[30] OPC Foundation. "OPC Unified Architecture Specification, Part 4: Services Release 1.04", 2017. Available at: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-4-services/,Web. 29 Aug. 2018

[31] Edward Curry, "Message-Oriented Middleware", National University of Ireland, 2004, Available at: https://pdfs.semanticscholar.org/98e1/90fd2ed9e15514f7f5d5ea3dbd8aeb382a9c.pdf,Web. 20 Sep. 2018

[32] Security Working Group of OPC Foundation. "Practical Security Recommendations for building OPC UA Applications", 2018, Available at: https://opcfoundation.org/wp-content/uploads/2017/,Web. 11 Nov. 2018

[33] OPC Foundation. "OPC Unified Architecture Specification, Part 2: Security Model Release 1.03", 2015. Available at: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-2-security-model/,Web. 03 May 2018

[34] Beckhoff New Automation Technology. “Hardware documentation for CX1020 / CX1030 Embedded PC”, 2014, Available at: https://download.beckhoff.com/download/document/ipc/embedded-pc/embedded-pc-cx/cx1020_hwen.pdf,Web. 06 Nov. 2018

[35] Beckhoff New Automation Technology. “Access to PLC runtime”, Available at https://infosys.beckhoff.com/english.php?content=../content/1033/tcopcuaserver/27021597842942731.html&id=,Web. 06 Nov. 2018

[36] Beckhoff New Automation Technology. “Manual CX2020, CX2030, CX2040”, Available at https://download.beckhoff.com/download/document/ipc/embedded-pc/embedded-pc-cx/cx2000_hwen.pdf,Web. 06 Nov. 2018

[37] Beckhoff New Automation Technology. “Manual TC3 OPC UA TwinCAT”, Available at https://download.beckhoff.com/download/document/automation/twincat3/TF6100_TC3_OPC-UA_EN.pdf,Web. 06 Nov. 2018

[38] Beckhoff New Automation Technology. “TS6100 | TwinCAT OPC UA Server”, Available at https://www.beckhoff.com/TS6100/,Web. 06 Nov. 2018

[39] Prof. Dr. Oliver Niggemann and Prof. Dr.-Ing. Jürgen Jasperneite “Industrie 4.0 Communication Guideline Based on OPC UA” , VDMA • Fraunhofer IOSB-INA • 2017, Available at https://industrie40.vdma.org/documents/4214230/20743172/Leitfaden_OPC_UA_Englisch_1506415735965.pdf/a2181ec7-a325-44c0-99d2-7332480de281, Web. 29 Jan. 2019

[40] Prof. Dr.-Ing. Reiner Anderl and Prof. Dr.-Ing. Jürgen Fleischer , “Guideline Industrie 4.0 Guiding principles for the implementation of Industrie 4.0 in small and medium sized businesses”, ISBN 978-3-8163-0687-0, 2016, Available at https://industrie40.vdma.org/documents/4214230/0/Guideline%20Industrie%204.0.pdf/70abd403-cb04-418a-b20f-76d6d3490c05, Web. 29 Jan. 2019

Appendix 1:

Partial declaration of variables and program body "PRG_READ_TC2_STRUCT_DIVERSOS()"

```
PROGRAM PRG_READ_TC2_STRUCT_DIVERSOS
VAR
    SessionConnectInfoArray     : ARRAY[1..10] OF ST_UASessionConnectInfo;
    nConnectionHdlArray         : ARRAY[1..10] OF DWORD;

    (* Declarations for UA_GetNamespaceIndex *)
    fbUA_GetNamespaceIndexArray : ARRAY[1..10] OF UA_GetNamespaceIndex;
    nNamespaceIndexArray        : ARRAY[1..10] OF UINT;

    (* Declarations for UA_NodeGetHandle *)
    fbUA_NodeGetHandleArray     : ARRAY[1..10] OF UA_NodeGetHandle;
    NodeIDArray                 : ARRAY[1..10] OF ST_UANodeID;
    nNodeHdlArray               : ARRAY[1..10] OF DWORD;

    (* Declarations for UA_Read *)
    fbUA_ReadArray              : ARRAY[1..10] OF UA_Read;
    stIndexRangeArray           : ARRAY[1..10] OF ARRAY [1..nMaxIndexRange] OF ST_UAIndexRange;
    nIndexRangeCountArray       : ARRAY[1..10] OF UINT;
    stNodeAddInfoArray          : ARRAY[1..10] OF ST_UANodeAdditionalInfo;

```

PRG_READ_TC2_STRUCT_DIVERSOS* | MAIN | GlobalSettings | Object Browser

```
PROGRAM PRG_READ_TC2_STRUCT_DIVERSOS
CASE iState OF
0:  (* idle *)
    IF bTest THEN
        bError      := FALSE;
        nErrorID    := 0;
        iIndiceBalancal:= iIndiceBalancal+1;  //incrementa indexador dos objetos
        IF (iIndiceBalancal>10 OR bHabilitaServer[iIndiceBalancal]=FALSE ) THEN
            iIndiceBalancal:=1;
        END_IF

        SessionConnectInfo.tConnectTimeout      := T#1M;
        SessionConnectInfo.tSessionTimeout      := T#1M;
        SessionConnectInfo.sApplicationName     := '';
        SessionConnectInfo.eSecurityMode        := eUASecurityMsgMode_None;
        SessionConnectInfo.eSecurityPolicyUri   := eUASecurityPolicy_None;
        SessionConnectInfo.eTransportProfileUri := eUATransportProfileUri_UATcp;

        stNodeAddInfo.nIndexRangeCount := nIndexRangeCountArray[iIndiceBalanca];
        stNodeAddInfo.stIndexRange := stIndexRangeArray[iIndiceBalanca];

        iState  := iState + 1;
    END_IF

1:  (* Open UA session *)

    IF (nConnectionHdlArray[iIndiceBalancal]=0) THEN  //tanto faz

        sOpcUaServerUrl:=sOpcUaServerUrlArray[iIndiceBalancal];

        fbUA_Connect(
            Execute             := TRUE,
            ServerURL           :=  sOpcUaServerUrl,
            SessionConnectInfo  := SessionConnectInfo,
            Timeout             := T#5S,
            ConnectionHdl       => nConnectionHdlArray[iIndiceBalancal]
        );

        IF NOT fbUA_Connect.Busy THEN
            fbUA_Connect(Execute := FALSE);

            IF NOT fbUA_Connect.Error THEN      (* session open *)
                iState := iState + 1;
            ELSE
                bError          := TRUE;
                nErrorID        := fbUA_Connect.ErrorID;
                nConnectionHdlArray[iIndiceBalancal]    := 0;
                iState          := 0; (* idle *)
            END_IF
        END_IF
    ELSE
        iState := iState + 1;   //sessão já existe - próximo passo
    END_IF

2:  (* GetNS Index *)
    IF (nNamespaceIndexArray[iIndiceBalancal]=0) THEN   //Se não existe o nNameSpaceIndex, cria
        sNamespaceUri:=sNamespaceUriArray[iIndiceBalancal];
        fbUA_GetNamespaceIndex(
            Execute         := TRUE,
            ConnectionHdl   := nConnectionHdlARRAY[iIndiceBalancal],
            NamespaceUri    := sNamespaceUri,
            NamespaceIndex  => nNamespaceIndexArray[iIndiceBalancal]
        );

        IF NOT fbUA_GetNamespaceIndex.Busy THEN
            fbUA_GetNamespaceIndex(Execute := FALSE);
            IF NOT fbUA_GetNamespaceIndex.Error THEN
                iState      := iState + 1;
            ELSE
                bError      := TRUE;
                nErrorID    := fbUA_GetNamespaceIndex.ErrorID;
                iState      := 62; (* valor iState identifica local onde o erro ocorreu - idle*)
            END_IF
        END_IF
    ELSE  //Se já existe, próximo passo
        iState:=iState + 1;
    END_IF

3:  (* Get Node Handle *)
    IF (nNodeHdlArray[iIndiceBalancal]=0) THEN
        NodeIDArray[iIndiceBalancal].eIdentifierType := eUAIdentifierType_String;
        NodeIDArray[iIndiceBalancal].nNamespaceIndex := nNamespaceIndexARRAY[iIndiceBalancal];
        NodeIDArray[iIndiceBalancal].sIdentifier     := sNodeIdentifier;
        fbUA_NodeGetHandle(
            Execute         := TRUE,
            ConnectionHdl   := nConnectionHdlArray[iIndiceBalancal],
            NodeID          := NodeIDArray[iIndiceBalancal],
            NodeHdl         => nNodeHdlArray[iIndiceBalancal]
        );

        IF NOT fbUA_NodeGetHandle.Busy THEN
            fbUA_NodeGetHandle(Execute := FALSE);
            IF NOT fbUA_NodeGetHandle.Error THEN
                iState      := iState + 1;
            ELSE
                bError      := TRUE;
                nErrorID    := fbUA_NodeGetHandle.ErrorID;
                iState      := 63; (* valor iState identifica local onde o erro ocorreu - idle*)
            END_IF
        END_IF
    ELSE
        iState      := iState + 1;
    END_IF

4:  (* UA_Read *)
    fbUA_Read(
        Execute         := TRUE,
        ConnectionHdl   := nConnectionHdlArray[iIndiceBalancal],
        NodeHdl         := nNodeHdlArray[iIndiceBalancal],
        cbData          := SIZEOF(nReadData),
        stNodeAddInfo   := stNodeAddInfoArray[iIndiceBalancal],  //última
        pVariable       := ADR(CacambasPesadasBalancasGeral[iIndiceBalancal])
        );

    IF NOT fbUA_Read.Busy THEN
        fbUA_Read( Execute := FALSE, cbData_R => cbDataRead);

        IF NOT fbUA_Read.Error THEN
            iState      := 0;   //volta pra processar próxima comunicação
        ELSE
            bError      := TRUE;
            nErrorID    := fbUA_Read.ErrorID;
            iState      := 64; (* valor iState identifica local onde o erro ocorreu - idle*)
        END_IF
    END_IF

5:  (* Release Node Handle *)
    fbUA_NodeReleaseHandle(
        Execute         := TRUE,
        ConnectionHdl   := nConnectionHdlArray[iIndiceBalancal],
        NodeHdl         := nNodeHdlArray[iIndiceBalancal]
    );

    IF NOT fbUA_NodeReleaseHandle.Busy THEN
        fbUA_NodeReleaseHandle(Execute := FALSE);

        IF NOT fbUA_NodeReleaseHandle.Error THEN
            iState      := iState + 1;
        ELSE
            bError      := TRUE;
            nErrorID    := fbUA_NodeReleaseHandle.ErrorID;
            iState      := 65; (* valor iState identifica local onde o erro ocorreu - idle*)
        END_IF
    END_IF

(* artifício para rastrear erros *)

62: (* close session -- error in 2*)
    iState      := 6;
63: (* close session -- error in 3*)
    iState      := 6;
64: (* close session -- errpr in x*)
    iState      := 6;
65: (* close session *)
    iState      := 6;
6:  (* close session default*)

    fbUA_Disconnect(
        Execute         := TRUE,
        ConnectionHdl   := nConnectionHdlArray[iIndiceBalancal]
    );

    IF NOT fbUA_Disconnect.Busy THEN
        fbUA_Disconnect(Execute := FALSE);
        bBusy := FALSE;
        IF NOT fbUA_Disconnect.Error THEN
            (* session closed *)
            iState      := 0;
            nConnectionHdlArray[iIndiceBalancal]:=0;
            nNamespaceIndexArray[iIndiceBalancal]:=0;
            nNodeHdlArray[iIndiceBalancal]:=0;
            IF NOT bError THEN
                bDone   := TRUE;
            END_IF
        ELSE
            bError          := TRUE;
            nErrorID        := fbUA_Disconnect.ErrorID;
            iState          := 0; (* idle *)

            nConnectionHdlArray[iIndiceBalancal]:=0;
            nNamespaceIndexArray[iIndiceBalancal]:=0;
            nNodeHdlArray[iIndiceBalancal]:=0;

        END_IF
    END_IF
END_CASE
```

Fig. 17 – Software developed for testing of OPC UA *client* features in TC3 (full content)