MCIP Client Application for SCADA in Iiot Environment

Nicoleta Cristina GAITAN[1,2]
[1]Faculty of Electrical Engineering and Computer Science, Integrated Center for Research, [2]Development and Innovation in Advanced Materials, Nanotechnologies, and Distributed Systems for Fabrication and Control (MANSiD)
[1,2]Stefan cel Mare University of Suceava
Suceava, Romania

***Abstract*—Modern automation systems architectures which include several subsystems for which an adequate burden sharing is required. These subsystems must work together to fulfil the tasks imposed by the common function, given by the business purpose to be fulfilled. These subsystems or components, in order to perform these tasks, must communicate with each other, this being the critical function of the architecture of such a system. This article presents a MCIP (Monitoring and Control of the Industrial Processes) client application which allows the monitoring and control of the industrial processes and which is object-oriented. As a novelty, the paper presents the architecture of the user object, which is actually a wrapper that allows the connection to Communication Standard Interface bus, the characteristics of the IIoT (Industrial Internet of Things) object and the correspondence between a server's address space and the address space of MCIP.**

Keywords—SCADA; OPC DA; OPC.NET; OPC UA; DDS

I. INTRODUCTION

SCADA (Supervisory Control and Data Acquisition) systems are those hardware/software systems that allow data acquisition from sensors or field devices used in the industrial process monitoring and control, and also allow the transmission of command/instructions to the remote field devices or actuators [1].

SCADA systems are usually distributed applications on a local network or WAN. The main elements of the SCADA architecture and security are presented in [2]. Usually in process control, the typical information architecture is hierarchically organized. A simplified model contains the following levels: process and business management; process control; management of field devices. These levels are not clearly defined, but the vertical communication up to the process component level is always necessary.

This communication requires solving the following requirements: to provide an adequate level of reliability; to comply with the time limits for delays; to support a diverse communication infrastructure; to use the standards specific for the producers in order to access the data from the process; the communication architecture must use open solutions (independent of the producers) for further development; to provide a uniform model for data presentation. An innovative idea is to solve the problem of system integration using a homogeneous architecture and a real-time process level.

The architecture must be based on the international standards for data exchange which allow data sharing from the devices placed on the hierarchical levels of the enterprises by the control and management systems of the processes. This architecture is a virtual level model which offers an overview of the underlying process level consisting of data units accessible randomly by means of a standardized and unified interface. As a result, the structure of the links becomes systematic and, more than that, the superior levels can be preserved.

The architecture must also ensure an optimum transfer of data based on a simple but generally accepted rule that the most important data must be transferred only once at any given time. To achieve this goal, the acquisition of the data contained by the process must be performed using appropriate communication technologies. This functionality is very important in order to achieve communication in the systems using remotely-controlled terminals spread over a large geographical area, such as heat and electricity meters.

Examples of such overall architectures were presented in [3] and [4] for the Classic OPC based on architecture for servers starting from standards [5] and [6] and in [7] based on a general architecture for servers named OPC Unified Architecture (OPC UA) [8].

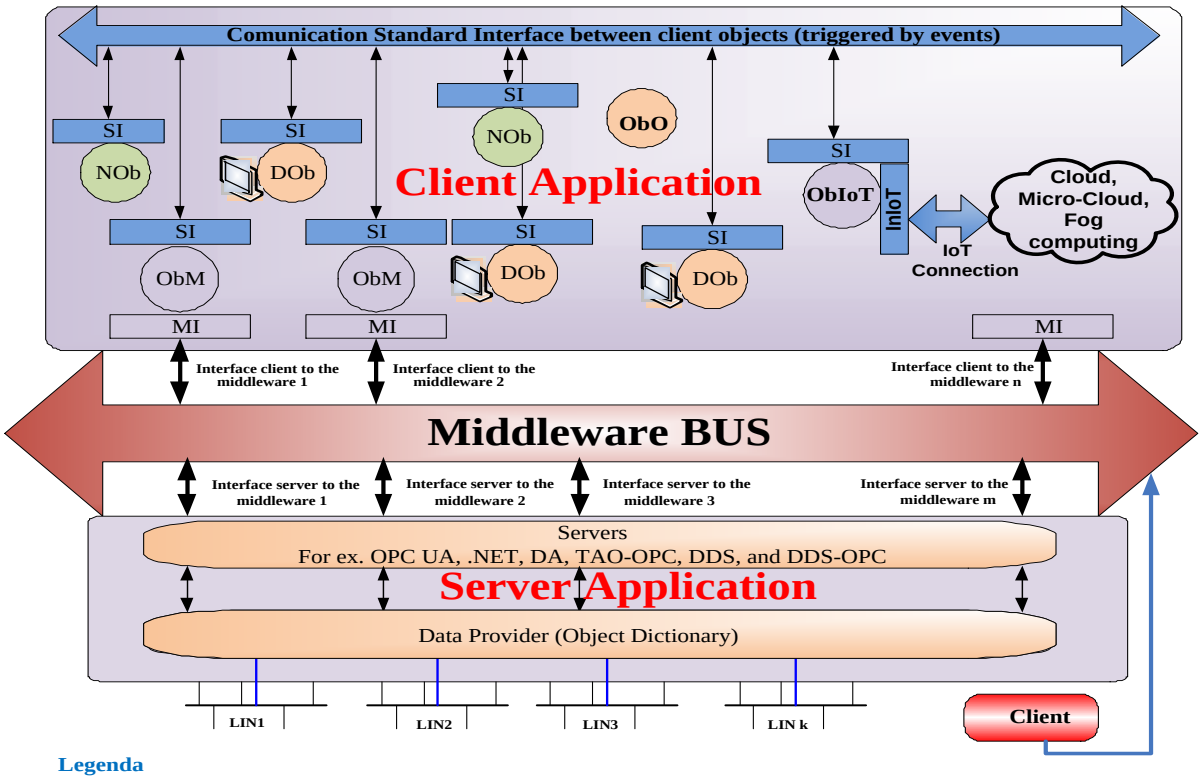
An idea related to the standardization of the data is the use of gateways which work in real time and which standardize the access to various local industrial networks that implement different algorithms [9][10].

The MCIP application, originally submitted in [11], is object-oriented, the objects implementing the standard interface of the client application which allows the communication between the application objects. In addition to this standard interface, each object may implement other interfaces for the communication with other applications (middleware interface) or user interaction (graphical interface). The acronyms MCPI (Monitoring and control industrial processes) in [11] and MCIP are equivalent.

Further, this article is structured as follows: Section II presents the client application and its objects, in Section III the processing triggered by events is presented, Section IV presents the IoT (Internet of Things) object, Section V presents the future development of the proposed solution. The conclusions are drawn in Section VI.

II. THE MCIP CLIENT APPLICATION AND ITS OBJECTS

This application was developed to allow the monitoring and control of the industrial processes. The MCIP application is an executable software module which is part of a complex architecture (see Fig.1). MCIP has the following types of objects: ornament objects (background or fonts for windows or objects), graphical objects (display values from other objects or expressions with values from other objects – window, scale, trend, button, input box, etc.), expression-type objects which produce value based on the evaluation of an expression that has as an input values from other objects, middleware-type objects which have two interfaces, one with the MCIP application's internal bus and one with a middleware bus (the bus has well defined API functions, see Fig. 1), a middleware interface specific for a server (OPC UA, .NET, Classic, TAO_OPC and DDS_OPC) and the Internet of Things-type objects such as DDS (ObIoT from Fig. 1). The TAO_OPC and DDS_OPC servers are based on a set of interface functions defined by the OPC DA 2.05 specification, while the middleware is specific (CORBA for TAO_OPC and DDS for DDS_OPC). The items marked in italics surpass the MCPI described in [11]. The middleware objects define the MCIP application's local address space, while the DDS server defines the MCIP application's global space IIoT.



Legenda

NOb - Normal Object
SI - Standard Interface
LIN - Local Industrial Network
DOb - Display Object
MI - Middleware Interface
InIoT - Industrial Internet of Things
ObM - Object interface to middleware
ObIoT - Object Internet of Things
ObO - Object Ornamental

Fig. 1. The architecture of the MCIP application

When designing the client application, the following important features were taken into account from the beginning:

- The MCIP application is fully object-oriented. By object we understand, in its most general meaning, a software module with internal functioning, which has a set of input sizes - binary or analogue, internal memory and a set of output sizes - binary or analogue, resulting from internal processing.
- The objects' space is called Project. In a project there can be several window-type objects, depending on the application's complexity.
- The mcip.exe application supports a single project. If more than one project needs to be implemented, more mcip.exe applications will be left running.
- Only within a project can all the objects communicate among themselves.
- Only the middleware-type objects, OPC (Classic, NET and UA), TAO_OPC (The ACE (Advanced Collaborative Environment) ORB), DDS_OPC (Distributed Data Services) and DDS can create the connection to the data servers, histories and alarms and events.

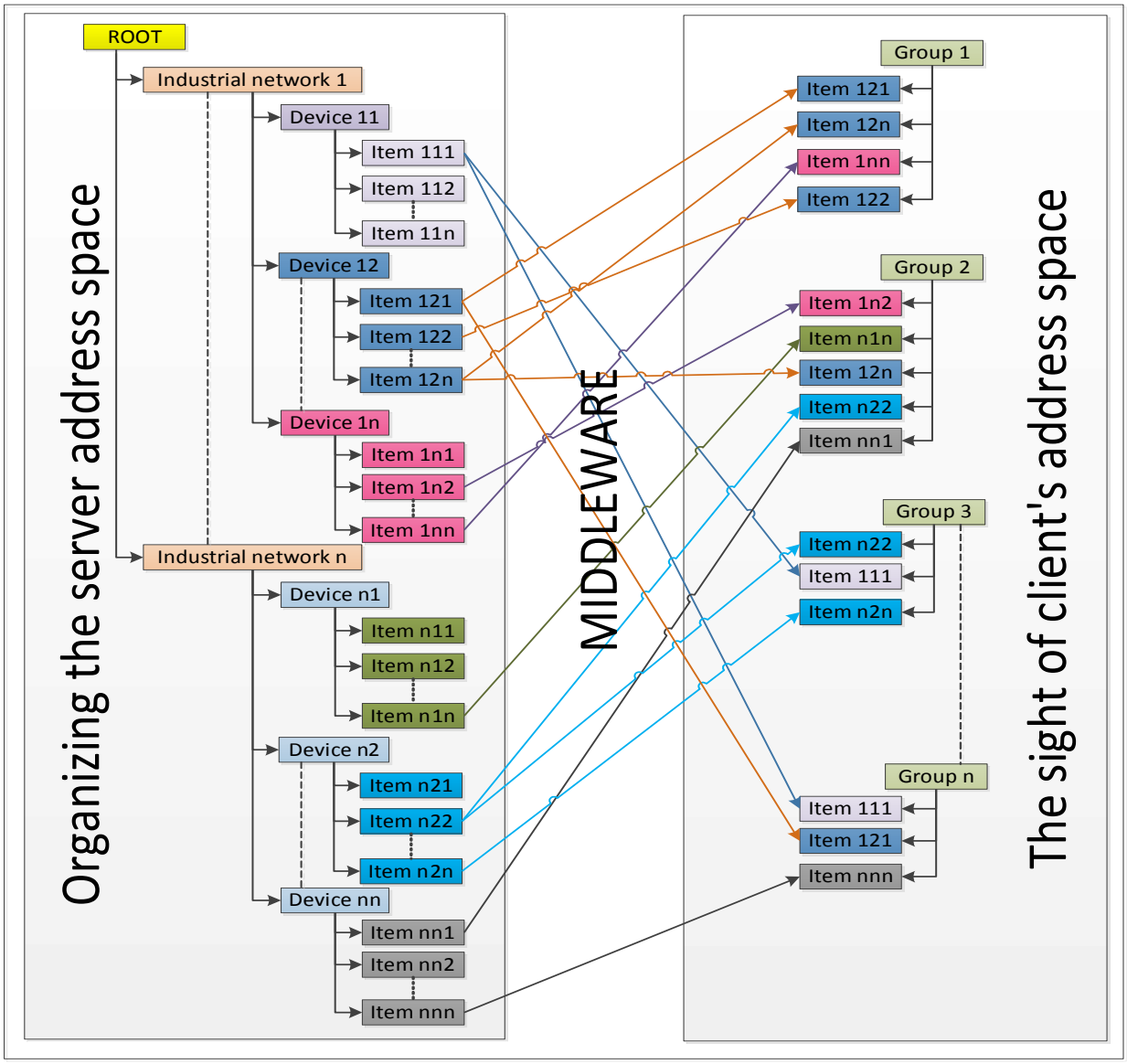


Fig. 2. The correspondence between the server's address spaces and MCIP

The middleware objects are designed to create groups with common features (only pressures, only temperatures, a complete installation, a control loop with parameters and the connected orders, etc.) or any type of group depending on the user's needs.

- A middleware object can be connected to a single server placed on the same computer as the application or on another computer in the network.
- A value exposed (supplied) by the server can be taken up by any number of local middleware objects (from the same computer) or placed at a distance.
- The MCIP application is secured with a set of passwords which allow only its use or both its use and modification.
- The MCIP application has two working modes, namely:
  - ✓ The *editing mode* when it is disconnected from the servers and it allows the implementation or modification of the project, and
  - ✓ The *execution mode*, when the application runs what it was designed to run.
- The mcip.exe application has a basic set of objects to which, by loading some dynamic *.dll type libraries, one can add other objects supplied by the producer or objects can be added on demand or programmed by the user. This is possible because the objects have a standardized interface.

A. *The objects of the MCIP application*

As shown in Fig. 1, objects always have a standard interface but they can also have other ones such as the interfaces with the servers described and a user-defined interface. From this point of view, we can distinguish the

following types of objects: a) general form, producer-consumer object; b) empty object (text or image); c) an object which is only a producer; d) an object which is only a consumer. The graphical objects always have an expression in the form of text or image in the window. For non-graphical objects, it is not mandatory for their expression in the window to be text or image (they are hidden). The software module attached to the object runs only when the trigger is activated which is achieved only on events (for example, changing the value of a quantity).The MCIP application consists of an ensemble made up of a project, objects and connections, the user's being to create, configure, and connect the objects among together. In this subsection, the objects made available by the application (the MCIP client) will be presented. A project is a group of independent or interconnected objects. The user creates projects (separate MCIP applications) in order to perform certain tasks. The user can open and close projects without affecting the other projects running in parallel. Each object will have among its resources the parameter setup window and other windows needed to display the object's information and configuration. It is possible for an object to have multiple functions (for example, an object which can calculate either the minimum or the maximum for a given input variable). The middleware-type objects are the only objects which allow the connection to the middleware servers. Because the architecture of the MCIP application is based on objects and on the connections between objects, once a middleware objects is created, any other object within a project will be able to connect to it. These objects can be graphic displays, control objects through which the human operator has access to the process values.

This is the usual mode of communication with the physical devices part of the process, but, if necessary, communication objects internal to the application can be developed (for serial communication, Ethernet, wireless – with specific interfaces). Each object has the "BaseObject" interface and can optionally have a middleware or a specific interface.

The entirety of a project's objects forms the project's address space which will always be displayed in the window named "Manager of objects". There is an exception here, namely that the graphical objects, those displaying a text or image, are not included in a project's address space being managed only as information display elements at window level (those objects which do not produce outputs and, therefore, to which other objects cannot connect).

The user will be able to create multiple windows in the same project and will be able to connect the objects from the same window or different windows but will not be able to connect objects belonging to different projects. The client application will allow the users to access take over and manage the data displayed by the server. The user will be able to view and browse the server's address space, creating groups which will contain a series of items available at this address space.

In other words, the customer will create his own vision of the address space exhibited by the server. This concept is illustrated in Fig. 2.

B. The user object

What is represented in Fig. 3 is an object adaptor (an instance of a class) to which a base object is attached which implements the standard interface), when created by the user. Under this form can the objects from the application's object directory be found (of the project) [11]. This is an artifice for the implementation of the connections between objects, because the object's common functionalities would have otherwise implemented themselves repeatedly in each base object's case. The adapter automatically attaches the client application whenever an object is created and, as a result, at the dll level which implements new base objects, only the standard interface must be implemented (CSI – Communication Standard Interface of the Object Client).

The directory of the user objects is a set of objects and subdirectories used to group the project's user objects in various categories. It will handle the management of the component user objects. It consists of:

- List of directories;
- List of user objects.

The connection. Objects can be interconnected, allowing signals to pass from one object to another. Thus, an object's data members can connect to the data members of another object. A connection is described by the following information:

- A data member/alias handle which has connections (that enter into an expression associated with a parameter, or to which another object's data member is connected (directly or through an expression), or which is displayed on a form (directly or through an expression);
- A list of objects to be notified when the above-mentioned data member/alias changes. Each listed item will be a structure containing an object handle and a handler list of data members or parameters to be notified for that object.

User object name. Each object must have a unique name within the project/directory (each object will belong to a project/directory). The name of the object will be indicated in the properties window, from where the parameters will be configured. A set of rules must also be defined and must be followed in order to give an object a name.

Alias. An alias for a data member is a generic name which can be used in a project even if the connection with other objects has changed (even if used a thousand times, there is no need to change the name in a thousand places in the project). It contains information such as:

- Data member handle;
- Alias name (optional). If a name for the alias is given, it creates a new instance of the data member. Afterwards, the alias can be configured with a different configuration from that of the original data member. The aliases are used to isolate the process from the

hardware changes. Thus, supposing that for a multichannel indicator, called IUM04, IUM04.Ch1 (Channel 1) has the alias Pressure and that everywhere in the process it is referred to through this alias instead of the data member, as IUM04.Pressure. Later, for some reason, the pressure is acquired on Ch2 (Channel 2). In this case, the alias will simply be changed to refer to IUM04.Ch2 instead of IUM04.Ch1. If only the parameters are changed and a name for the alias is not given, all the connections towards the data member will take into account these parameters.

- Description;
- The following information depends on the data type of the data member for which the alias is given:
  - ✓ Numeric: linear scaling from one domain to another; deviation – in order not to take into account the insignificant variations of the data members (if the difference between the last and the current value does not exceed the deviation, it is considered that the amount did not change); a forced value – the data member will have this value no matter what happens in the process. Usually, it is used when a sensor breaks or when a sensor is being repaired, or when a PLC receives erroneous signals.
  - ✓ Logic: The possibility of reversing the logic signal.
  - ✓ Text: It has no additional particularity.

Attribute. The generic term of attribute will be used for any parameter, data member or alias of a base object. Where this term appears, when it does not refer to all the properties, what it refers to will, in principle, be listed (for example, through an attribute only the parameters and the data members can be determined in certain contexts). Attribute = data member = properties.

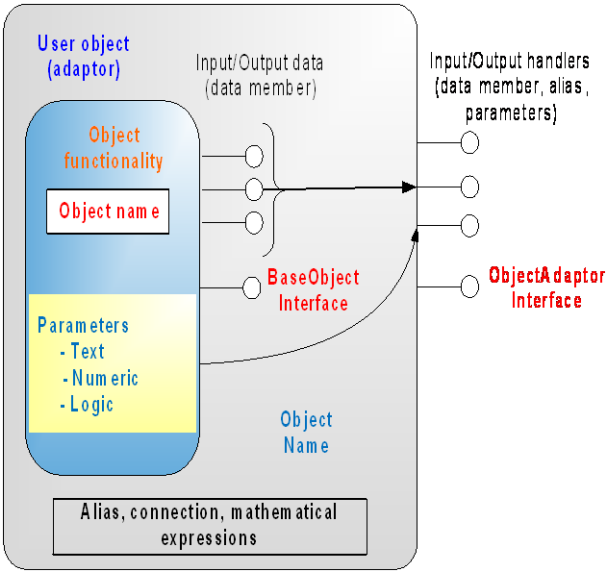


Fig. 3. The architecture of the user object

C. The interface of the object implemented in dll

To access the object implemented in the dll a standard interface is defined. The term object described above in the terminology section refers to an object in the project which is actually an adapter for the object in dll (because, for an object, additional information is needed – aliases, filters, connections – which do not need to be doubled, tripled in dlls). A standard interface will be created (or a base class containing all the virtual methods) to be implemented by all the objects defined as independent components in dlls and by the possible objects implemented in the application. The code implementation has been done in C#. For the fast access to the information about the data member and about the reading and writing methods of the data members, a Hash table is used. The objects within the hash tables are name_data_member–object_type_MemberInfo pairs. For each data member exposed by the by the current object, there will be a pair in the hash table.

III. Processing Triggered by Events

At process level, the code is only executed when an event is activated. The characteristics referring to the events are as follows:

- The functioning of the client application is entirely driven by events.
- A particular code sequence is executed only when an input variable changes.
- The execution of the code in an infinite loop is avoided in order to save processor time.
- The update loops are avoided (the events which generate themselves recursively).
- Each object remains inactive until an event occurs on one of its connections.
- When an input signal changes, the object processes the value according to the internal logic (given by the type of object).
- The objects emit events only when the processing result has changed.

This application's approach will consume much less processor time than a solution based on waiting in a loop until the input signal changes.

IV. IoT Objects

Several current technologies have revolutionized everyday life. One of them is the Internet that has led to a new era of information, available for everyone. Other technologies, such as Radio-Frequency Identification (RFID), or Wireless Sensors Network increased the ability to communicate among things. Internet Protocol Version 6 (IPv6) removed the address depletion problem existing in the Internet Protocol (IPv4). In this context, it is only natural that the next wave in the development of the Internet does not refer to people, but to interconnected smart devices. The mobile revolution, with more than 5 billion smart phones connected to various mobile networks, provides the population with the possibility to access the Internet.

DDS RTS (Data Distribution Service for Real-Time Systems) is a middleware standard based on the publish/subscribe paradigm for distributing data between heterogeneous applications developed by OMG consortium [12]. An important feature of this protocol is that it has facilities for implementing QoS (Quality of Service) parameters in order to achieve real-time performance. It is also data centric and allows the anonymous dissemination of information. Due to real-time facilities, this protocol is used in critical systems to the detriment of OPC based solutions. An interesting comparison between DDS and OPC can be found in [13]. There are several open source implementations of the DDS standard [14], namely OpenDDS and OpenSpice, and commercial implementations, for example, RTI-DDS.

The DDS standard can be used in two ways: as an object in the MCIP application and as a "DDS server" type application which instantiates the data provider. For both versions, the IDL interface is used, in which the topics to be published are defined. For this reason, data can be transmitted between the MCIP applications, between the DDS server applications or between the MCIP applications and the DDS servers. In MCPI, the DDS object actually creates an interface between the DDS topics defined in IDL and in the MCIP internal communication bus and in the DDS server an interface is created between the API provided by the data provider and the DDS topics defined in IDL. The DDS will define the IoT workspace.

The OpenDDS object is currently in the development and implementation stage. This object will expose the address space of the DDS domain in the MCIP environment on which it can connect. Each instance of the OpenDDS objects can connect to only one DDS domain in terms of available QoS parameters. In order to develop this object, the OpenDDS implementation of the DDS standard was chosen because it is an open-source solution and it is developed based on the TAO middleware.

V. Conclusion

In this article, we have detailed aspects related to the MCIP client. Thus, new MCIP objects were defined, the adapter structure for user objects was presented, and a new object was added which allows it to connect to the IIoT-type (Industrial Internet of Things) applications. The object is based on the DDS protocol from OMG because of the existence of a free implementation called OpenDDS.

VI. Future Work

In the future, other IoT-type objects based on protocols will be taken into account such as XMPP (Extensible Messaging and Presence Protocol, AMPQ (Advanced Message Queuing Protocol), MQTT (Message Queuing Telemetry Transport (MQTT), OPC UA (OPC Unified Architecture), REST (Representational State Transfer) and, CoAP (Constrained Application Protocol).

Acknowledgment

This paper was partially supported from the project „Integrated Center for research, development and innovation in Advanced Materials, Nanotechnologies, and Distributed Systems for fabrication and control", Contract No. 671/09.04.2015, Sectorial Operational Program for Increase of the Economic Competitiveness co-funded from the European Regional Development Fund..

References

[1] S. A. Boyer, SCADA: supervisory control and data acquisition, 4rd ed., International Society of Automation, ISBN-10: 19360070962009, June 15, 2009.

[2] R. Krutz, Securing SCADA Systems, ISBN-10: 0-7645-9787-6, Wiley, 2006.

[3] Nicoleta-Cristina GĂITAN, Vasile Gheorghiță GĂITAN, Ștefan Gheorghe PENTIUC,Ioan UNGUREAN, Eugen DODIU, Middleware Based Model of Heterogeneous Systems for SCADA Distributed Applications, pag. 121-124, ISSN: 1582-7445 e-ISSN: 1844-7600, Advances in Electrical and Computer Engineering Volume 10, Number 2, 2010.

[4] Iwanitz, Franz și Lange, Jurgen. OPC - Fundamentals, Implementation, and Application, third rev. Ed. [ed.] Softing. 3. Munich : Huthig, ISBN 3-7785-2904-8, 2005.

[5] OPC Foundation - DA 3.0. 2003. OPC Data Access Custom Interface Standard Version 3.0. 2003.

[6] OPC Foundation- DA 2.05. 2002. OPC Data Access Custom Interface Standard Version 2.05. 2002.

[7] Lange, Jurgen, Iwanitz, Frank și Burke, Thomas J. 2010. OPC From Data Access To Unified Architecture. [ed.] Softing. Berlin : VDE VERLAG GMBH,ISBN 3-978-3-8007-5, 2010.

[8] OPC Foundation: Home Page, https://opcfoundation.org/

[9] V.G. Gaitan, N. C. Gaitan, I. Ungurean, A flexible acquisition cycle for incompletely defined fieldbus protocols, ISA Transactions, Volume 53, Issue 3, Pages 776-786, ISSN 0019-0578, http://dx.doi.org/10.1016/j.isatra.2014.02.006, May 2014.

[10] Gaitan, N. C. Real–time Acquisition of the Distributed Data by using an Intelligent System. Electronics and Electrical Engineering.–Kaunas: Technologija 8 (2010): 104, 2010.

[11] Vasile Gaitan, Valentin Popa, Nicoleta Cristina Gaitan, Mihai Gabriel Danila, A scalable Human-Computer Interaction (HCI), Proceedings of ED-MEDIA 2008 – World Conference on Educational Multimedia, Hypermedia & Telecomunications, , pag. 1522-1527, ISBN 1-880094-62-2, Viena, Austria, (http://www.editlib.org/), June 30- July 4 2008.

[12] Object Management Group, http://www.omg.org/ H. Perez, J.J. Gutierrez, "A survey on Standards for real-time distribution middleware" Journal ACM Computing Surveys, vol. 46, issue 4, March 2014.

[13] Comparison of OPC and DDS – RTI, https://www.rti.com/docs/RTI_DDS_and_OPC.pdf.

[14] Best-Practices Data-Centric Programming: Using DDS to Integrate Real-World Systems, https://www.rti.com/docs/DDS_Best_Practices_WP.pdf.