Proceedings of the 18th International Conference on System Theory, Control and Computing, Sinaia, Romania, October 17-19, 2014

Transparent Interaction of SCADA Systems Developed over Different Technologies

Ioan UNGUREAN, Nicoleta Cristina GAITAN, Vasile Gheorghita GAITAN
Faculty of Electrical Engineering and Computer Science
Stefan cel Mare University of Suceava
Suceava, Romania
ioanu@eed.usv.ro, cristinag@eed.usv.ro, gaitan@eed.usv.ro

***Abstract*— Supervisory control and data acquisition (SCADA) systems are usually distributed applications designed and developed by using a middleware technology. In this paper, it is proposed a solution for interoperability of SCADA systems that are based on different middleware technologies. The proposed solution allows the interconnection of SCADA system based on the following middleware technologies: OPC DA, OPC .NET, OPC UA, TAO (The ACE ORB – an open-source implementation of CORBA standard), and OpenDDS (an open-source implementation of the DDS protocol). The proposed solution is a software application that allows the creation of middleware objects in order to connect to the data servers. In the application framework, it is allowed the interconnection of the tags exposed by the middleware objects (tags are acquired from data servers through a middleware) directly or by a math expression. The proposed solution is scalable in the sense that it can be added new software modules for other types of middleware in addition to those listed above.**

Keywords—SCADA; OPC DA, OPC .NET, OPC UA, TAO, 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 monitoring and control of industrial process, 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 are the followings [2]:

- The human operator - that monitor the industrial process via the SCADA system.
- HMI (Human-Machine Interface) - that presents the information acquired from the industrial process in a graphical manner. This is a software module that can be executed on a computer or on a dedicated device, eventually with touch-screen. Furthermore, the human operator can use this module to send commands to the remote actuators.
- MTU (Master Terminal Unit) – that is the master unit in the master/slave architecture. This module acquires information from the fieldbuses [3] and transmits them to the HMI modules.
- RTU (Remote Terminal Unit) – that is the slave unit in the master/slave architecture. This is a remote device that acquires information from the industrial process and transmits them to the MTU.

RTU communicates with MTU throughout the fieldbuses [3]. MTUs can be embedded dedicated devices or PC software applications. The MTU modules distribute the data to the HMI modules in the Internet/extranet/intranet via the standard middlewares. The most-used middleware technologies are those based on OPC specifications [4] (OPC DA [5], OPC .NET [6], OPC UA [7]), and those based on CORBA standard [8][9], DDS (Data Distribution Service) protocol [9], or AMQP (Advanced Message Queuing Protocol) protocol [10].

Currently, there are a large number of SCADA system manufacturers [11] that use one or more middleware technologies listed above. In this paper, we propose a simple solution to exchange data between these SCADA systems based on different middleware technologies for data distribution in intranet/extranet/Internet.

Further, this article is structured as follows: Section II presents the main middleware technologies used to develop SCADA systems, in Section III it is presented the solution proposed for interoperability of SCADA systems, and Section IV presents future development of the proposed solution. The conclusions are drawn in Section V.

II. Middleware technolgies for SCADA sytems

In order to transmit data between MTU modules and HMI modules through heterogeneous networks, the SCADA systems use standards-based middleware systems. The most-used middlewares are those based on the OPC specifications that are developed and sustained by OPC Foundation [4]. The use of standard interfaces of software components, supported by several manufacturers, is the main motivation of the OPC Foundation. Currently, the OPC Foundation provides three middleware architectures (defined by standard specifications) for the developing of distributed applications for monitoring and control of industrial processes, with extension for MES and ERP applications, namely: Classic OPC DA (based on DCOM technology from Microsoft) [5], OPC .NET - initially called Xi Express interface (based on Windows Communication Foundation from Microsoft) [6], and OPC UA -Unified Architecture (based on SOAP and Web services) [7]. Classic

ISBN 978-1-4799-4602-0 ©2014 IEEE

OPC DA specifications have been generally accepted for many years as the most popular industry standard among developers and users of SCADA applications. The most manufacturers of applications like HMI (Human-Machine Machine Interface), SCADA (Supervisory Control and Data Acquisition), and DCS (Distributed Control System) based on the PC, provides a client and/or server with OPC interfaces for their products. In order to benefit of the security features (authentication, authorization and encryption) of the WCF (Windows Communication Foundation) technology, there are defined OPC .NET specifications. These specifications are defined for .NET platform and can transport the data in the Internet (there are not limited within a local network as classical OPC specifications). The possibility to expose not only pure data from the process but also alarms, historical data, and commands to web services, has led to a completely new approach, namely OPC Unified Architecture specification [7]. This specification defines the management of data from process, alarms, historical data, and advanced functions in a single unified address space. In addition to these new properties - platform independent and unified data management - OPC UA introduces numerous other properties, which include scalability, security on unauthorized access, precautions regarding data loss and support for complex data structures [7]. Example of SCADA application based on OPC specification, in addition to commercial solutions [11], can be found in [12] and [13]. Furthermore, on OPC Foundation site [4] is published a complete list of manufacturers of OPC based SCADA solutions.

CORBA (Common Object Request Broker Architecture) is a middleware standard based on the client/server paradigm to distribute data between heterogeneous applications in terms of programming language and operating system used. Specifications were developed by OMG consortium [14] consisting of active companies in the industry. For real-time distribution, there are defined RT-CORBA specifications that can achieve a deterministic access to shared resources and can apply multiple scheduling policies for multithreading applications [8]. There are several open source implementations of CORBA standard [9], namely TAO (The ACE ORB), PolyOBB or MICO, and commercial implementations, for example [9], VisBroker, ORBExpress and e*ORB. Although this is currently not widely used, CORBA is still a reliable solution for critical systems in the detriment OPC-based systems [17]. For example, CORBA is still used for data distribution in LHC Experiments' Control Systems [18]. Example of SCADA application based on CORBA implementations can be found in [15] and [16].

DDS (Data Distribution Service for Real-Time Systems) is a middleware standard based on the publish/subscribe paradigm to distribute data between heterogeneous applications developed by OMG consortium [14]. 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 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 [19]. There are several open source implementations of DDS standard [9], namely OpenDDS and OpenSpilce, and commercial implementations [9], for example, RTI-DDS. Example of SCADA application based on DDS implementations can be found in [20] and [21].

III. PROPOSED SOLUTION

In this paper, it is proposed a solution for interoperability of SCADA systems and distributed real-time and embedded systems developed with different technologies in terms of middleware used for data distribution. Therefore, to achieve this goal, it is proposed a framework, which will further be named MIOF (Middleware Inter-Operability Framework). The architecture of solution selected in order to implement the MIOF application is based on middleware objects and connections between these objects. Each object encapsulates a specific functionality depending on de middleware on which is based. Each middleware object has a set of parameters (through which the object can be defined and configured) and a set of tags or data members (which behaves like input/output points). Data members are actually tags that can be read and/or written to the data server on which the middleware object is connected. Logical architecture of such an object is shown in Fig. 1.

Through a standard interface defined in the MIOF application, tags exposed by different middleware object can be interconnected. This interface works on publish/subscribe paradigm. A tag may publish the value that he received through the specific middleware, and on this value can subscribe several tags that work as inputs. It should also be taken into account the data types of each tag and to make a conversion where is necessary and possible. Tags that subscribe to a value may apply different math operations on them. Therefore, in order to determine the input value, a tag can subscribe to one or more output tags and the obtained values can be used in a math expression that is evaluated at each change of a tag that is used in the math expression. In the defining operation of connections between the tags of middleware objects, it must take into account the access attributes of each tag (read only, write only, or read-write). In addition to this attribute, each tag has the following attributes: error code (similar to quality defined in classic OPC specifications), timestamp (when the value was produced), data type (logical, numeric, or text), and the value of the tag. Fig. 1 presents the logical architecture of a middleware object. As can be seen in Fig. 1, there is a data server that retrieves data from the process and can send various commands to actuators. These data are distributes by means of a middleware system middleware (OPC DA, OPC UA, OPC NET, TAO, or OpenDDS). Internal functionality of the server deals with management of communication with devices connected on fieldbuses. In order to communicate with these devices, it may be used different converters (e.g. RS232-RS485, USB-CAN) or server can be executed on embedded devices, which have direct connectivity to fieldbuses. Middleware objects take the address space of the server and expose it in the MIOF application. An example of the address space exposed in MIOF is shown in Fig. 1 (the address space is specific to the data server used).

At the MIOF application level, the code of an object is executed only when an event occurs or when a published value changes its value.

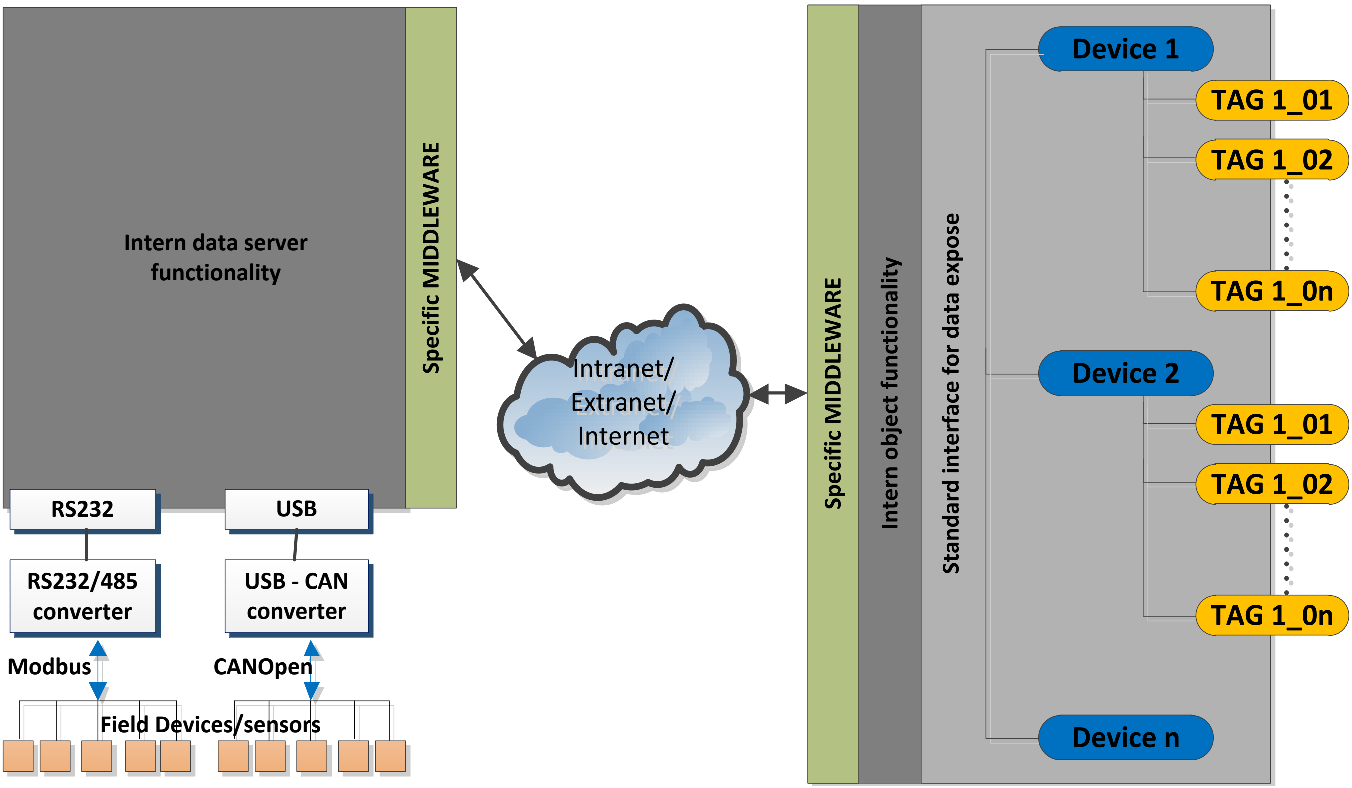

Fig. 1. Logical architecture of an middleware object.

MIOF application characteristics related to events are the following:

- Application functionality is driven entirely by events.
- Middleware objects emit events when they received from the data server a value of a tag.
- It is avoided the loop connection of the tag in order to avoid recursive generation of the events.
- Objects remain inactive until an event occurs on one of its connections.
- When an input tag is changed, the object will send the changed value to the data server on which is connected.
- Middleware objects allow the browse operation of the address space of server they connect to.
- The middleware object will perform the necessary operations in order to update only the tags that have at least one subscription (it do not update data that is not consumed by anyone).

This operating approach of the application consumes much less CPU time than a solution based on pending in the loop the change of the input signals (changing / updating tags from the data servers). The interconnection way of objects is presented in Fig. 2. It can be noted that a tag can subscribe to multiple tags that are placed in an arithmetic expression to obtain the final value.

The application is designed and developed in C#, and the standard interface is represented by a base object (implemented by a class named BaseObject) from which are derived the middleware objects. The base object defines methods used to save and restore specific parameters of the object in an XML, functions used to perform the browse operation over the address space, a function to read a list of tags, a function to write a list of tags, and a delegated that is called by the object when it is needed to update (publish) the tags values in the MIOF environment. Each object is developed as a library, which exposes a class derived from the BaseObject class. In the MIOF application, it is used an implementation artifice in the form an adapter object (an instance of a class) that is attached to an object middleware (that implements the standard BaseObject interface) by the user when it create a new object. This method avoids the redundant implementation of the common functionalities at the level of each middleware object derived from the base object. Within this adapter, the management of the connections of the attached middleware object is performed. A connection is described by the following information:

- A handle of a tag/data member that has connections (which is connected to at least a data member of another middleware object in the MIOF environment directly or through a math expression).
- A list of objects that must be notified when the data member is changing. Each item in the list will be a structure that contains an object handle and a list of tags handlers that must be notified by the object.

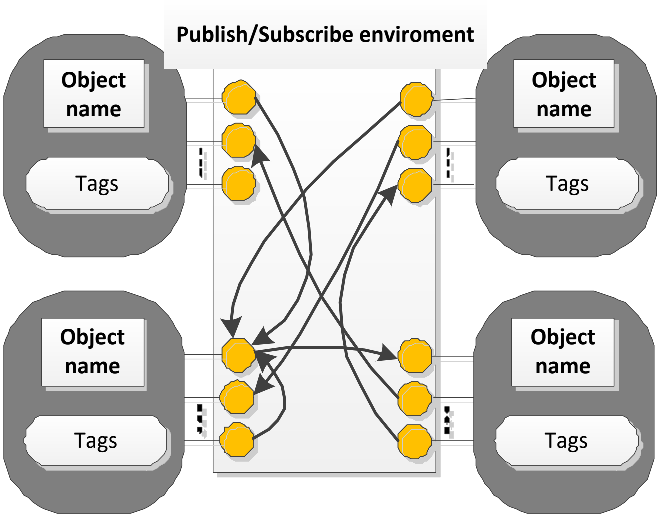

Fig. 2. The interconnection way of middleware objects.

The adapter contains a list of connections for the attached middleware object. This method simplifies the integration of new types of middleware objects. At this moment, there are implemented middleware objects for the OPC DA, OPC. NET, OPC UA, and TAO, and the middleware object for the OpenDDS implementation of the DDS standard in the development stage.

A. *The OPC DA Object*

The purpose of the OPC DA object is to perform the connection between the servers based on classical OPC DA specifications and the MIOF application. Therefore, this object implements two interfaces: interface defined for the objects from MIOF application and the interface for OPC DA data server. In terms of internal functionality, this object implements a wrapper that performs the interconnection of the two interfaces of the OPC DA object. In order to implement the OPC DA interface were used RCW (Runtime Callable Wrappers) components that manage the COM methods calls in .NET based applications. These methods are free components distributed by the OPC Foundation

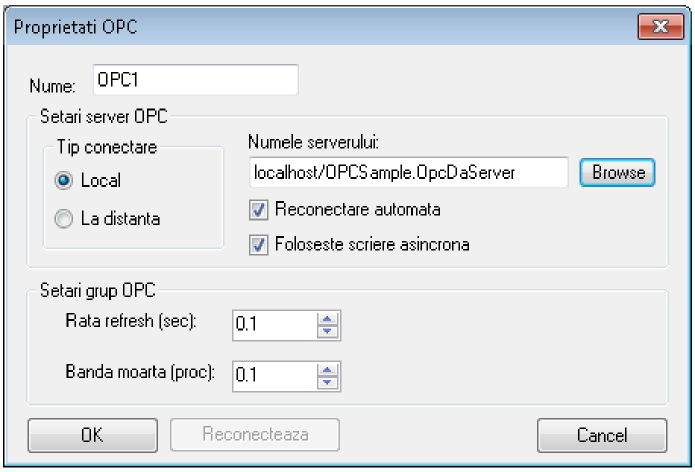

Fig. 3. The proprities of the OPC DA object.

When an OPC DA object is created in the MIOF environment, it must be set the specific properties of the object, namely: OPC DA server on which it connects, the deadband of the group created by this object, and the refresh rate of the OPC DA group. After setting these parameters, the object will connect to the OPC DA server and create a group with the properties that have been set, group that contains no tag.

When a connection is made between every member of the OPC DA object (tag from the OPC server on which it is connected) and another object from MIOF (may be even other OPC DA object), then the tag will be added to the group associated to the OPC DA object. When it is deleted a connection that is made with a data member of the OPC DA object, then it will delete the item that corresponds to that date member (only if the data member is not used to make other connections). There will be no limit of OPC DA object that can be created in the MIOP application. Fig. 3 presents the window used to set the properties of the OPC DA object.

B. *The OPC .NET Object*

The aim of the OPC.NET object is to perform communication with the servers based on OPC.NET specifications and MIOF application. When it is created an OPC.NET object, it must be set the specific properties of the object, namely: the host where runs the NET OPC server on which it connects, deadband and refresh rate for the list created by this object in the server, communication protocol used (TCP or HTTP), port used, and authentication information such as user name and password associated (if the server needs this data to allow the access). After setting these proprieties, the object will connect to the OPC.NET server and create a list (structure defined by the OPC.NET specifications) with properties that have been set (the list contains no tag).

When a connection is made between every tag of the OPC.NET and another tag from MIOF application (may be even other OPC object) then the tag will be added to the list. There is no limit of OPC.NET object that can be created in the MIOF application. In Fig. 4 is presented the window used to set the properties of the OPC .NET object. Fig. 4 presents the window used to set the properties of the OPC DA object.

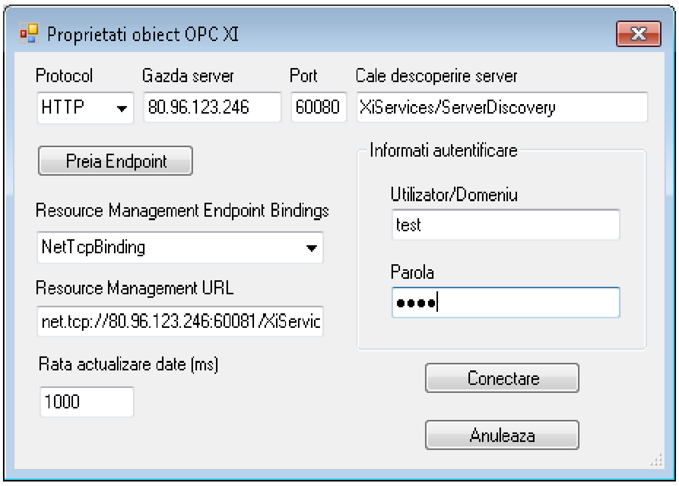

Fig. 4. The proprities of the OPC.NET object.

C. *The OPC UA Object*

The aim of the OPC UA object is to perform communication with the servers based on OPC UA specifications and MIOF application. When it is created an OPC UA object, it must be set the specific properties of the object, namely: host where the OPC UA server runs, deadband for subscription created on this object on the OPC UA server (structure defined in the OPC UA specifications), the refresh rate of the associated subscription, the protocol used for communication (TCP or HTTP) and authentication data used to access the OPC UA server. After setting these parameters, the object will connect to the OPC UA server and create a subscription with the properties that have been set. When is performed a connection between a tag of the OPC UA object(tag from the address space of the OPC UA server) and another object from the MIOF application (may be even an OPC UA object) then it will add the tag to the associated subscription. There is no limit of OPC UA object that can be created in the application MFIOP. OPC UA object can create a single subscription in the OP UA server. If it is wanted the creation of another subscription with different parameters, then it must create a new OPC UA object. Fig. 5 presents the window used to set the properties of the OPC DA object.

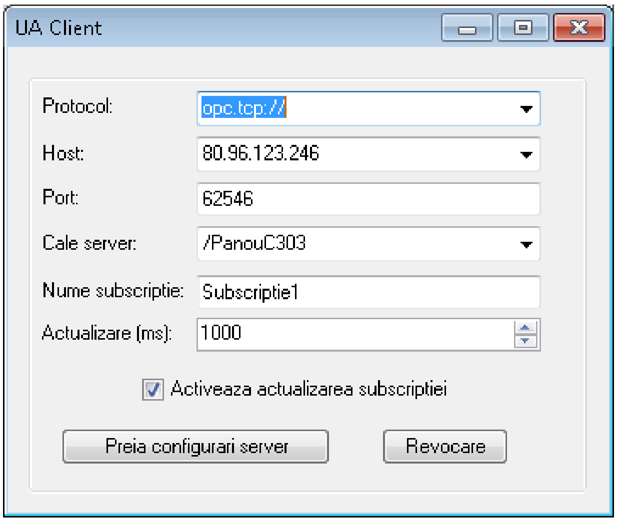

Fig. 5. The proprities of the OPC UA object.

D. *The TAO Object*

TAO object connects the MIOF application with the TAO servers presented in [22]. When a TAO object is created, it must set the properties of the object, namely the host where the TAO server runs, the protocol used for communication (GIOP, SSLIOP, and ZIOP) deadband and refresh rate of the group created in the server by the object. Moreover, in this case, there is no limit of TAO objects that can be created. It can be created multiple TAO objects that connect to the same server, but with different parameters for the associated group (deadband and refresh rate). Because the TAO middleware is developed in C++, it was developed a wrapper in order to call the functions of TAO middleware from the .NET based application. Fig. 6 presents the window used to set the properties of the OPC DA object.

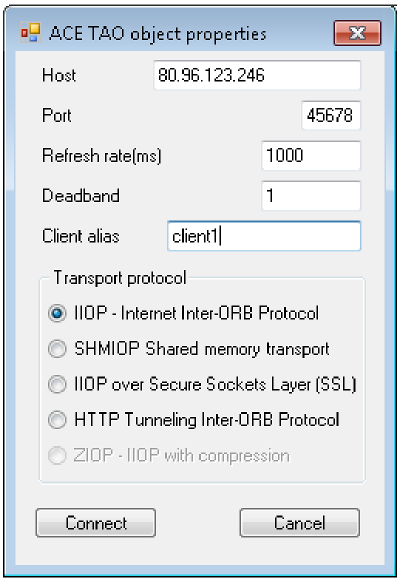

Fig. 6. The proprities of the TAO object.

E. *The OpenDDS Object*

OpenDDS object is currently in development and implementation stage. This object will expose in the MIOF environment the address space of the DDS domain on which 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, it was chosen the OpenDDS implementation of the DDS standard because it is an open-source solution and is developed based on the TAO middleware.

IV. Future Work

As future work, it can be developed objects that will take the address space of the MIOF environment, and they can distribute it through a specific middleware. Thus, it can be created OPC UA, OPC. NET and OPC UA objects that work as servers, and they expose the address space of the MIOF environment. By this method, the HMI applications can connect to these objects to retrieve data and display them in a specific graphical manner.

V. Conclusion

This article has presented a solution for interoperability of SCADA system developed over different technologies in terms of middleware used to distribute data acquired from monitored and/or controlled industrial processes. The solution allows the interconnection tags directly or by applying math expressions. Simplifying, the proposed solution operates on the publish/subscribe paradigm in the sense that the data produced by a tag can be consumed by more tags. In the proposed MIOF application, it can be easily introduced new objects that can retrieve data through a specific middleware or other methods such as service-oriented applications. Furthermore, it can be developed objects that read data directly from fieldbuses and publish them in the MIOF environment.

ACKNOWLEDGMENT

This paper was supported by the project “Sustainable performance in doctoral and post-doctoral research PERFORM–Contract no. POSDRU/159/1.5/S/138963”, project co-funded from European Social Fund through Sectorial Operational Program Human Resources 2007-2013.

REFERENCES

[1] S. A. Boyer, SCADA: supervisory control and data acquisition, 4rd ed., International Society of Automation, 2009.

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

[3] V.G. Gaitan, N. C. Gaitan, I. Ungurean, “A flexible acquisition cycle for incompletely defined fieldbus protocols,” ISA Transactions, vol. 53, Issue 3, pp. 776-786, May 2014.

[4] OPC Foundation, https://opcfoundation.org/

[5] F. Iwanitz and J. Lange H. Fachverlag, OPC—Fundamentals, Implementation and Application, Huthig, 2006.

[6] The Express Interface delivers secure and reliable real-time and historical data communication, http://expressinterface.com/.

[7] W. Mahnke , S.-H. Leitner , M. Damm, OPC Unified Architecture, Springer Publishing Company, 2009.

[8] D. C. Schmidt, B. Natarajan, A. Gokhale, N. Wang, and C. Gill, “TAO: A Pattern-Oriented Object Request Broker for Distributed Real-time and Embedded Systems,” IEEE Distributed Systems Online, vol. 3, no. 2, February 2002.

[9] H. Perez, J.J. Gutierrez, "A survey on Standards for real-time distribution middleware," Journal ACM Computing Surveys, vol. 46, issue 4, March 2014.

[10] S. Vinoski, "Advanced Message Queuing Protocol," Internet Computing, IEEE , vol.10, no.6, pp.87-89, Nov.-Dec. 2006.

[11] HMI Software & SCADA Software Manufacturers, http://www.automation.com/suppliers/automation-product-manufacturers/product-category/hmi-software-scada-software

[12] A. Girbea, C. Suciu, S. Nechifor, F., Sisak, "Design and Implementation of a Service-Oriented Architecture for the Optimization of Industrial Applications," Industrial Informatics, IEEE Transactions on , vol.10, no.1, pp.185-196, Feb. 2014

[13] S. Back, S. B. Kranzer, T. J. Heistracher, T. J. Lampoltshammer, "Bridging SCADA systems and GI systems," Internet of Things (WF-IoT), 2014 IEEE World Forum on, pp.41-44, March 2014

[14] Object Management Group, http://www.omg.org/

[15] S. S. Andrade and R. J. Macêdo, “Real-Time Component Software For Flexible And Interoperable Automation Systems,” Anais do XII Congresso Brasileiro de Automática (XII CBA), 1, pp. 3014-3019, 2006.

[16] S. S. Andrade and R. J. Macêdo, “Using Real-Time Components to Construct Supervision and Control Applications,” 8th Brazilian Workshop on Real-Time Systems - Work-in-Progress Paper. Curitiba - PR, 2006.

[17] D. C. Schmidt, “Future of CORBA for Distributed Real-time & Embedded Systems,” ICALEPCS 2014, http://www.dre.vanderbilt.edu/~schmidt/ICALEPCS.ppt, 2014.

[18] C. Gaspar, "An Overview of the LHC Experiments' Control Systems," Proceedings of ICALEPCS 2013, San Francisco, CA, USA, 2013

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

[20] P. L. Martínez, L. Barros, J. M. Drake, “Design of component-based real-time applications,” Journal of Systems and Software, Vol. 86, Issue 2, pp. 449-467, February 2013.

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

[22] E. Tatulescu, A. V. Smolenic, and V. G. Gaitan. "An Architectural Model of a CORBA based Data Server for SCADA Systems," International Journal of Academic Research, vol.5, no. 2, 2013.

Proceedings of
2014 18th International Conference on System Theory, Control and Computing (ICSTCC)

(Joint conference SINTES 18, SACCS 14, SIMSIS 18)

October 17 - 19, 2014

Editors:
Mihaela Hanako Matcovschi
Lavinia Ferariu
Florin Leon

Technical Co-sponsor:

Institute of Electrical and Electronics Engineers - Control Systems Society

Co-organizers:

Gheorghe Asachi Technical University of Iasi
Faculty of Automatic Control and Computer Engineering

University of Craiova
Faculty of Automation, Computers and Electronics
Automatic Control Research Centre

Dunarea de Jos University of Galati
Faculty of Automatic Control, Computers, Electrical and Electronics Engineering

SaC5 Invited Session, Bucegi 1

Interactions in Complex Systems

Chair: Craus, Mitica	Gheorghe Asachi Tech. Univ. of Iasi
Co-Chair: Ungurean, Ioan	Stefan cel Mare Univ. of Suceava
Organizer: Hulea, Mircea	Gheorghe Asachi Tech. Univ. of Iasi
Organizer: Teodorescu, Horia-Nicolai	Romanian Acad.

10:50-11:10, Paper SaC5.1

A Bio-Inspired Model to Care for Casualties in a Disaster (I)

Butincu, Cristian Nicolae	Gheorghe Asachi Tech. Univ. of Iasi
Craus, Mitica	Gheorghe Asachi Tech. Univ. of Iasi
Gavrila, Augustin Ionut	Gheorghe Asachi Tech. Univ. of Iasi

11:10-11:30, Paper SaC5.2

Towards an Inclusive Parkinson's Screening System (I)

Geman, Oana	Stefan Cel Mare Univ. of Suceava
Sanei, Saeid	Univ. of Surrey
Chiuchisan, Iuliana	Stefan Cel Mare Univ. of Suceava
Graur, Adrian	Stefan Cel Mare Univ. of Suceava
Procházka, Aleš	Inst. of Chemical Tech. Czech Republic
Vyšata, Oldřich	Inst. of Chemical Tech. Czech Republic

11:30-11:50, Paper SaC5.3

Transparent Interaction of SCADA Systems Developed Over Different Technologies (I)

Ungurean, Ioan	Stefan Cel Mare Univ. of Suceava
Gaitan, Nicoleta Cristina	Stefan Cel Mare Univ. of Suceava
Gaitan, Vasile Gheorghita	Stefan Cel Mare Univ. of Suceava

11:50-12:10, Paper SaC5.4

Study of the Long-Term Effect of STDP in Areas of Spiking Neurons (I)

Hulea, Mircea	Gheorghe Asachi Tech. Univ. of Iasi

12:10-12:30, Paper SaC5.5

An Ontology of Human Walk for Autonomous Systems (I)

Luca, Ramona	Inst. of Computer Science, Romanian Acad. Iasi Branch
Bejinariu, Silviu Ioan	Inst. of Computer Science, Romanian Acad. Iasi Branch
Teodorescu, Horia-Nicolai	Romanian Acad

12:30-12:50, Paper SaC5.6

Characterizing the Attractors of Chaotic Systems by a Direct Measurement Method (I)

Teodorescu, Horia-Nicolai	Romanian Acad
Hulea, Mircea	Gheorghe Asachi Tech. Univ. of Iasi
Cojocaru, Victor	D. Ghitu Inst. of Electronic Engineering and Nanotechnologie

SaD1 Invited Session, Miorita

Advances in Automotive Control - I

Chair: Popescu, Dan	Univ. Pol. of Bucharest
Co-Chair: Onea, Alexandru	Tech. Univ. of Iasi
Organizer: Lazar, Corneliu	Gheorghe Asachi Tech. Univ. of Iasi
Organizer: Petrescu, Marian	Continental Automotive Romania SRL Iasi

15:00-15:20, Paper SaD1.1

Modeling for Deployment Techniques for Intra-Car Wireless Sensor Networks (I)

18th International Conference on System Theory, Control and Computing

Joint Conference SINTES 18, SACCS 14, SIMSIS 18

17 - 19 October 2014, Sinaia, Romania

Mr. Ioan Ungurean
Stefan cel Mare University of Suceava
Suceava, Romania
720229 Suceava
Romania

November 3, 2014

Dear Mr. Ioan Ungurean,

On behalf of the Program Committee, it gives me great pleasure to invite you to participate in *the 18th International Conference on System Theory, Control and Computing ICSTCC 2014* which will be held at the Rina Sinaia Hotel, Sinaia, ROMANIA, during October 17 - 19, 2014.
The *ICSTCC 2014* is technically co-sponsored by the IEEE Control Systems Society (CSS). The Proceedings will be published in the *IEEE Xplore Digital Library* and will be submitted for indexing in the *Conference Proceedings Citation Index*.

Your paper submitted to the *ICSTCC 2014* has been accepted for presentation by the conference. As indicated in the notification letter sent to you about your paper's acceptance, at least one author of your paper must attend the conference to present the paper. We hope that you will participate in this scientific meeting.
Acceptance of your paper for presentation does not, in any way, financially oblige *ICSTCC 2014* for the expenses incurred by you to travel and attend the conference. If you have any questions, please contact us at icstcc2014@ac.tuiasi.ro.

WARNING: Depending on your citizenship, you may require visa to enter Romania. For additional information about visa and travel authorization, please visit the following website: http://www.mae.ro/en/node/2040

Thank you in advance for your participation. I look forward to seeing you in Sinaia.

Sincerely,
Prof. Mihail Voicu, General Chair of the ICSTCC 2014

Accepted Paper details:

Ioan Ungurean, Nicoleta Cristina Gaitan, Vasile Gheorghita Gaitan, "Transparent Interaction of SCADA Systems Developed Over Different Technologies." Scheduled for presentation on Saturday October 18, 2014, 11:30-11:50 hrs.

ICSTCC 2014

Certificate of Attendance

This certificate is awarded to

Ioan Ungurean

from

Stefan cel Mare University of Suceava
Romania

for attending the
18th International Conference on
System Theory, Control and Computing
Sinaia, Romania
October 17-19, 2014

Mihail VOICU
General Chair of ICSTCC 2014

IEEE CSS