REMOTE EXECUTION FUNCTION BLOCKS FOR DISTRIBUTED SYSTEMS

Oana ROHAT[1], Dan POPESCU[2]

Remote execution allows a distributed controller to have access to higher computational power and knowledge than provided by its own resources. Using the benefits gained by the standardization of distributed applications based on the IEC61499 reference model, the paper presents a function block remote execution system that can be used for a large variety of applications like process optimization, image processing, plant risk analysis etc. The system was designed as a web application using open technologies like PHP, Java and IEC 61499.

Keywords: remote execution, distributed systems, function block programming, software integration

1. Introduction

The benefits gained from connecting a plant to a supervisory control center are no longer questioned by process control engineers. Next generation applications must conform to the industrial needs to support enterprise integration, remote supervisory control and even web-based systems able to support cooperative work [1][2]. This trend comes to fulfill the needs of the process engineers for increased efficiency, productivity, flexibility and reliability.

Remote execution allows a distributed controller to have access to higher computational power and knowledge than provided by its own resources. This way algorithms that require great processing effort (from areas like process modelling, optimization, advanced control based on artificial intelligence techniques, plant risk analysis, image processing etc.) can be stored and executed on a remote server and use a simple network connection to have access to process parameters and send their results.

The IEC 61499 standard [3] provides a methodology for implementing platform-independent applications for distributed systems. Its support in defining communication objects and designing a system based on an application-centric approach provide a strong basis in the implementation of distributed control algorithms residing on remote nodes. As the standard gains more adopters and

[1] Eng., Dept. of Automation and Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: oana.rohat@gmail.com
[2] Prof., Dept. of Automation and Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail:dan_popescu_2002@yahoo.com

was already successfully implemented in a variety of applications [4][5], it is time to take the next step towards its web integration.

A web-based execution system gives the possibility of easy reconfiguration, diagnosis and optimization. While in most typical scenarios an application is run on local centralized controllers, the engineering process may be more efficient and more flexible with a distributed control architecture in which certain process-intensive functions are executed remotely. That is because the process control engineer can have access to a variety of algorithms that can be used without the need of a controller upload, just by using the existing communication link. Such a system can be used for different applications, from slow process control (agriculture, biology etc.), building management systems, plant monitoring, complex algorithms testing, model learning, detecting optimal control parameters etc. These applications most not have hard real-time constraints as the reaction time depends both on the network performance and server load.

This work presents the design and implementation of a web-based library that allows the remote execution of IEC 61499 function blocks using the FBDK (Function Block Development Kit) environment. The library provides access to reusable algorithms that can be configured for on-line execution with the results sent to a remote device.

The rest of this paper is organized as follows: Section 2 presents the system architecture, components and functions. Section 3 details the implementation aspects from the technology, communication and distributed execution points of view. In Section 4 a system prototype is evaluated. Conclusions are summarized in Section 5.

2. System architecture

The system architecture illustrated in Fig. 1 presents the system structure and defines the transfer of information between the corresponding subsystems. The system structure follows a multilevel hierarchical model that allows the decomposition of the implemented functions.

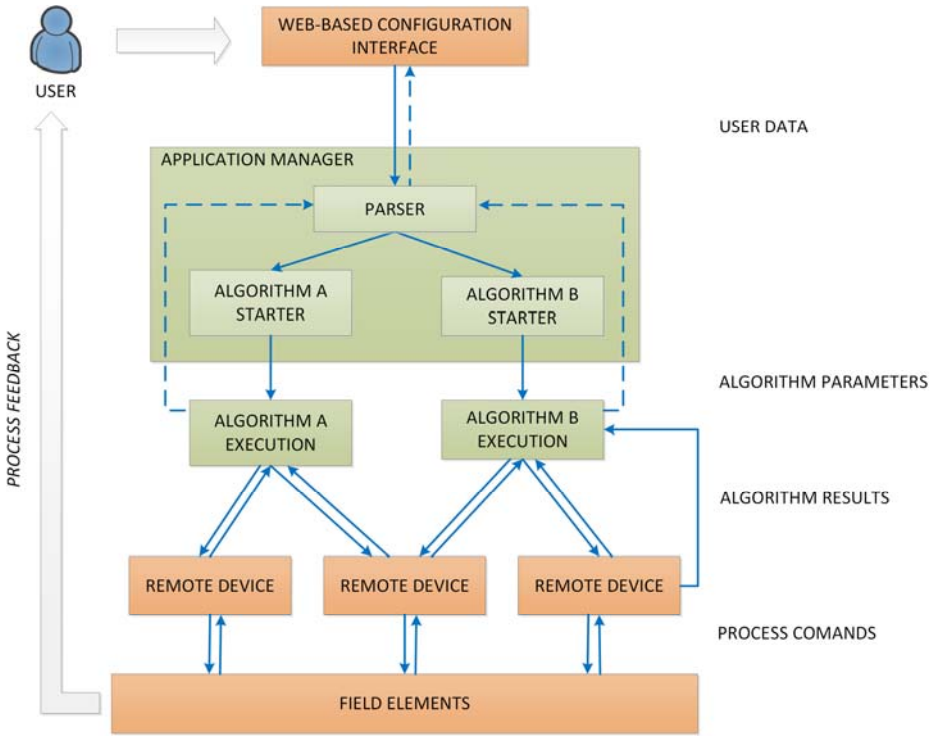

Fig. 1. System architecture

The highest level is represented by the web-based configuration interface that allows the user interaction and involvement in the control process. This level controls the execution of the algorithms and sends to the next level the algorithm parameters as they were entered by the user. These parameters include algorithm identification, execution data and the remote device addresses.

The application manager has the highest complexity as it has to implement the interface with the higher level, by the use of a parser block, and the specific algorithm starter blocks that ensure the connection to the execution environment. The algorithm parser is responsible for the correct interpretation of the received user data, the algorithm identification and for the generation of a specific handling process for every requested algorithm. Each such process is illustrated as an algorithm starter block and implements the algorithm start and stop control functions and the transmission of parameters to the next level.

The algorithm execution level receives the input parameters from the application manager in a format that it can understand, executes the algorithm and sends the results using the communication interface to the level below.

The remote device level is represented by the field controllers and implements a specific communication interface that allows it to access the

algorithm results so that it can use them in its control logic to command the field devices from the lowest level. Information regarding the plant response is sent back to the algorithm execution through the remote device.

In the current implementation, the plant response or the algorithm results are not accessible from the web interface. The user can observe the process reaction to a certain algorithm configuration and react accordingly by adjusting algorithms parameters in the web interface or by selecting a different algorithm. In addition future extension of the system will include making the feedback and algorithm results available also in the user interface.

The two upper levels in the system architecture are general levels, as their functionality does not change depending on the user interaction. The next two levels are specific levels as an algorithm is executed or a remote device is selected according to the user's options. Also, these last two levels are designed as generic, reusable components meaning that any algorithm can be used for any remote device considering they have matching communication interfaces.

3. Implementing the Web Execution System

3.1. Technologies used

The system is based on open technologies allowing the easy integration and interoperability of its components. Also, using open technologies provides the benefit of a wider community of users and developers that can quickly offer answers and opinions for good solutions.

The web application uses HTML for interface configuration and design and PHP for forms and user session handling, database access and data encapsulation into XML. PHP scripts are easy to write, easy to use and can be embedded into HTML. Using text file operation functions like fread() and fwrite() from PHP allows the programmer to encapsulate information regarding parameter values into XML files so that they can be transmitted to the next level [6].

The application manager was implemented using Java. This object-oriented platform independent technology allows developing secure, reliable applications and has inherently integrated network capabilities that ease the effort of connecting to the adjacent levels [7]. In addition to that it provides the tools for multithreaded programming that are needed for user requests and algorithm execution management. Java is also the technology behind the IEC 61499 FBDK editor [8] used for developing and running the algorithms.

The main technology behind the lower levels, algorithm execution and remote device, is the IEC 61499 standard because it provides the tools for developing distributed applications, uses an open representation format and is not dependable of the implementation platform. These levels use the IEC 61499 standard for algorithm implementation, execution and communication. Since the

IEC 61499 was designed for developing distributed applications, an important role is played by the communication objects [9] described below.

3.2. Communication

3.2.1. XML over TCP Protocol

The communication between the web interface and the application manager uses the TCP protocol. The TCP protocol is a reliable transport protocol that guarantees the correct order of the data transmitted over the underlying networking layers.

The web application collects the algorithm parameters entered by the user as input in the web page and sends the parameters using a TCP socket that connects to the application manager. The parameters entered by the user in the web application forms are encapsulated into an XML message. This message is sent to the Java application implemented at the application manager level using the TCP protocol, the message is parsed and the parameter values are extracted. This transfer was depicted in Fig. 2.

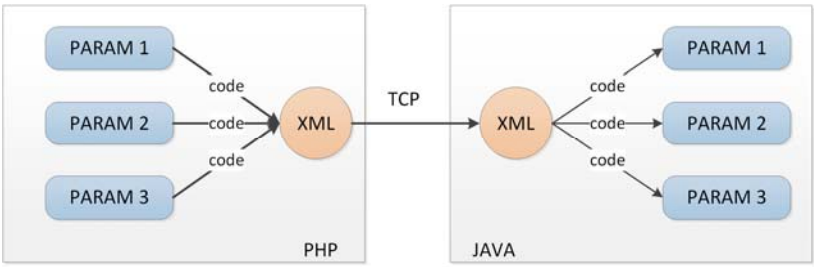

Fig. 2. Message encapsulation and transfer

3.2.2. Publish/Subscribe messaging

Sending the parameters from the application manager and the algorithm execution levels, as well as from the algorithm execution level to the remote device level is done using the Publish/Subscribe and Client/Server mechanisms. At the application manager level we have a publisher implemented as a software object using Java. It is responsible for transmitting the parameters from the user interface. At the algorithm level, we have implemented both a subscribe function block for the connection to the upper level and a server function block that sends the results to the plant device. Both these functions are based on the IEC 61499 standard.

The standard IEC 61499 standard defines two generic communication objects that implement standard messaging protocols: Publish/Subscribe for unidirectional point-to-multipoint communication and Client/Server for bidirectional point-to-point communication [10]. These objects can be used as they are or can be further developed for implementing specific industrial

communication protocols like Modbus, Profibus, CAN, OPC etc. The choice of the communication pattern was done according to the network requirements. The publish/subscribe pattern is recommended for use in local communications while the client/server for data transfer over the Internet.

In both mechanisms the link between the two blocks is done by the ID parameter. It represents the network address and must have the same value on each side. Parameters are sent from the publisher to the subscriber and bidirectional to and from the client and server in the same order they were received as input. While in the unidirectional communication the publisher is not informed if the subscriber received the data, nor if it was received correctly, the client server mechanism runs over TCP so data tracking and retransmission are available.

3.3. Distributed execution system

The system was designed following the Model Driven Architecture (MDA) presented in [10] that proposes an application-centric approach that looks at the system as a whole and afterwards distributes functionalities on different devices. This methodology was created to improve the design, development and implementation process by developing generic applications and applying them to specific platforms. This means that instead of defining the system specifications regarding the hardware components at the beginning of a project the application is designed as a Platform Independent Model (PIM) that implements only the system functionality. Following this, a Platform Definition Model (PDM) captures the devices and communication infrastructure so that in the end a Platform Specific Model (PSM) defines the assignment of system components to devices, input and output modules and addresses etc.

In designing the distributed system we first imagined and tested the system functionality as a whole, following a PIM, in a centralized approach based on the IEC 61499 standard. This method allows the process control engineer to implement and test all system requirements without depending on a specific hardware platform. A model of the centralized application for a specific algorithm was illustrated in the upper side of Fig. 3. Based on the role of each function block and the devices involved, the components of the centralized system were divided between three modules: execution parameters area, the algorithm area and the commands area.

In a standard application this would be followed by cutting the connections between the different modules and adding standard communication objects in the points where the cutting occurred so that afterwards these modules can by assigned to specific devices in the system configuration to obtain the PDM. Fig. 3 illustrates how the centralized application was split up between three modules: user interface, algorithm execution and process. As the standard does

not provide web HMI interface, after splitting the three areas and adding the communication blocks we further transformed the execution parameters software module with a web-based configuration interface. To ensure the proper connection to algorithm area in a way that would not interfere with the remaining two modules, we designed an application manager that would "translate" the interaction between the user interface and the algorithm execution. In order to have a minimum impact on the system model, this application manager was represented as a generic object that needs no additional reconfiguration when applied on a different algorithm.

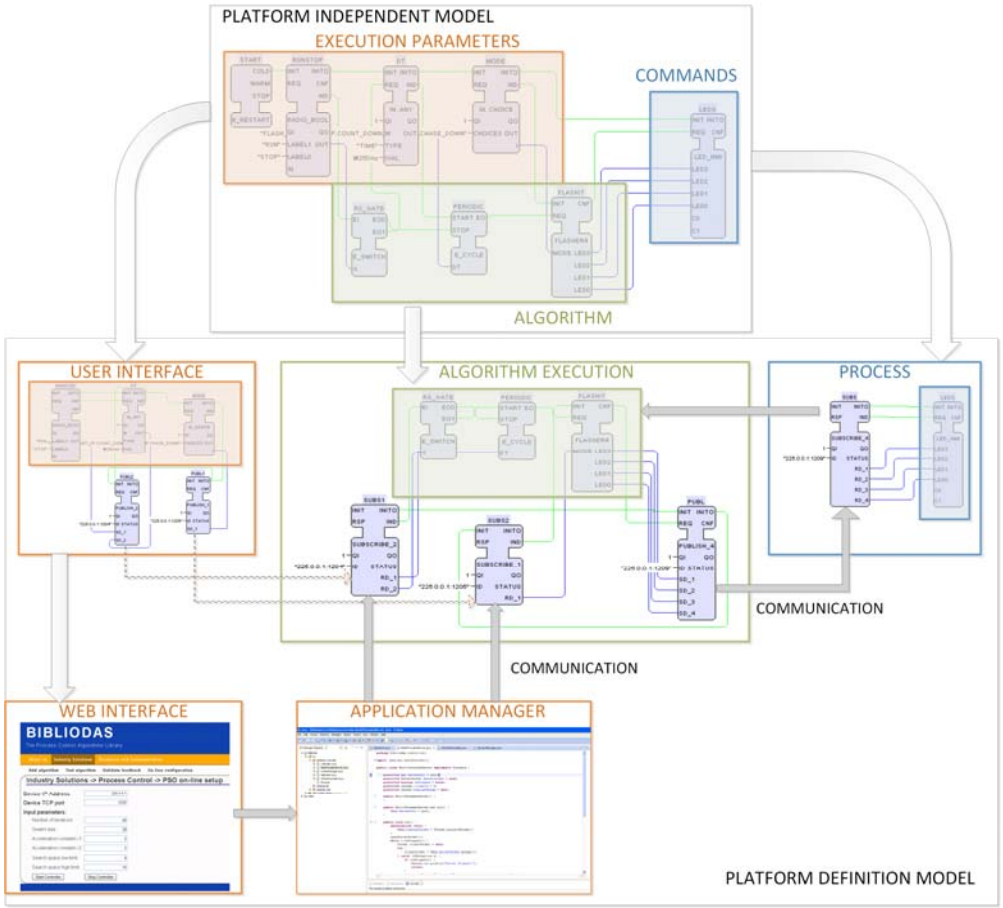

Fig. 3. Applying the model Driven Architecture (MDA) in the design of the distributed system

The PSM is obtained by the user when selecting an algorithm and applying specific parameters values and device addresses, thus defining a configuration that is applicable only to a specific process.

This modelling approach must be applied for every available algorithm. Even if the resulting system does not follow the true MDA modelling approach as the web interface is not based on the same programming standard as the algorithm execution and commands modules and an additional object is inserted, it still benefits from its advantages. That is because it provides a generic structure that can be applied on any algorithm with minimum reconfiguration effort and on any process that ensures the proper communication interface.

3.4. Handling communication problems

Using the Internet for remote control rises several problems related to the reliability and speed of the transmitted data. As it is based the TCP protocol, this ensures a minimum level of data transmission protection by using error detection and packet sequencing mechanisms that enable the packet retransmission in case of packet alteration or loss. Still, the retransmission and path induced latencies lead to uncertain delays. This may affect the control system performance and even lead to the destabilization of the process [11].

The FBDK editor provides Publish/Subscribe and Client/Server communication function blocks that can detect network problems and report them through the Status parameter. This allows us to implement delay-handling or timeout mechanisms that would deal with network errors or loss of communication situations.

A presentation of different network control algorithms for delay handling is available in [11]. The selection of a specific network control methodology must be done according to the particularities of the process. For example, the robust control methodology can be applied for both linear and non-linear control systems and does not require a priori information of the probability distributions of the network delays. Considering t_{er} the delay between the execution system and the remote device, t_{re} the delay between the remote device and execution system the network delay can be expressed by the following equation (adapted from [11]):

$$\begin{aligned} t &= \frac{1}{2}(t_{\min} + t_{\max}) + \frac{1}{2}(t_{\max} - t_{\min})a \\ &= (1-c)t_{\max} + act_{\max} \end{aligned}, \tag{1}$$

Where $a \in [-1, 1]$ is a delay variation factor, c is a constant in the interval [0,1/2] specific to the process and t represents the delay between the execution system and the remote device, either at data transmission or reception, and is bounded in the interval $[t_{min}, t_{max}]$.

In the frequency domain, the delay from (1) can be approximated with:

$$e^{-ts} = e^{-(1-c)t_{\max}s} e^{-act_{\max}s}$$
$$\approx \left(\frac{1-s(1-c)t_{\max}/2}{1+s(1-c)t_{\max}/2}\right)\left(\frac{1-sact_{\max}/2}{1+sact_{\max}/2}\right) \quad (2)$$

The second term of equation (2) is considered simultaneous multiplicative perturbation that can be represented as:

$$\frac{1-sact_{\max}/2}{1+sact_{\max}/2} = 1 + \frac{ct_{\max}s}{1+ct_{\max}s/3.465}\Delta = 1 + W(s)\Delta \quad (3)$$

Where Δ is the perturbation function and W(s) is a weight factor. The robust controller for the closed loop system is designed based of these factors as can be seen in Fig. 4. A more accurate representation can be obtained by modelling also the delay between the actuator and the field sensor to the remote device. As this network has different characteristics, these delays will also have different values for the a and c parameters, as well as for the delay interval.

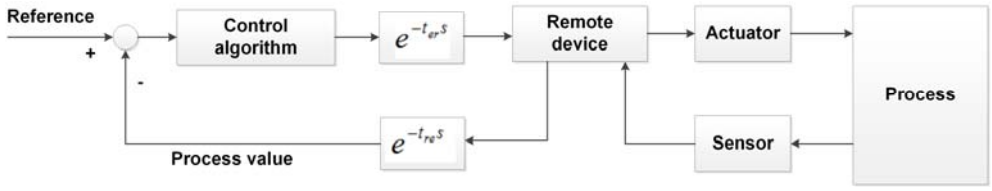

Fig. 4. System configuration based on the robust control methodology

4. System evaluation

A prototype based on the FBDK environment was evaluated using the system configuration presented in Fig. 5. This prototype aims at proving the system's usability, reliability and ease of reconfiguration.

The system configuration includes three computers: one with a web browser for user interface access, one where resided the web application server that also is responsible for executing the algorithm and one simulating a remote process device. The platform independency of the IEC 61499 standard allows us to use a remote computer running an FBDK application instead of a real process. These computers are connected in a local network.

For evaluating the system functionality we added two algorithms to the web library: Flasher, an example of a simple distributed system presented in [9], and PSO, an algorithm that can be used for example in the optimal tuning of the PID control method.

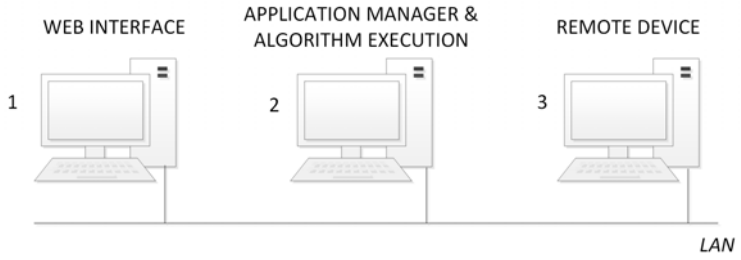

Fig. 5. System evaluation configuration

The system evaluation was conducted based on these use cases:

1) One user configures the execution on one algorithm on one device

The use case was tested on the PSO algorithm. To use the algorithm for remote execution a user must login to the library web-page using a standard web browser and select a desired algorithm following the Industry Solution/Process Control menu. The user specifies the IP and port needed for the connection to the remote device and then enters the algorithm parameters. Afterwards he can start or stop the remote execution of the algorithm. Leaving a field blank in the web interface or entering incorrect data type leads to an error message on the screen. The user must then enter the requested parameter and retry to start the algorithm. A status massage is available to show the current state of the algorithm. Closing the web page does not imply stopping the algorithm. The user can modify the parameter values at runtime but he can't change the IP or port of the remote device without stopping the algorithm execution.

For the communication with the application manager the system automatically generates a port number known by both components. This is used in combination with the application manager IP.

2) One user configures the execution of two different algorithms on one device

Each time a user logs in, he receives a session ID that helps the management of his running algorithms. This way, he can view the status of the configured algorithm, the current values of the input parameters and how much time passed since the algorithm was started. For testing this functionality, on the same computer one user started both the Flasher and the PSO algorithm and was able to see their status, their uptime and access the algorithm configuration page from the On-line configurations menu. The parameters of each algorithm could be modified without affecting the other one. Stopping an algorithm has no effect on the other algorithm as the communication with the controller is based on the UDP protocol that has no link status notification mechanism. This means the device

will continue to use the last known value of the algorithm parameters, even if the algorithm was stopped.

3) Two users configure the execution of the same algorithm on different devices

This use case was implemented by accessing the PSO algorithm configuration page in the web execution system from the computer 1 as in the earlier situation, and also from the computer 2. In this case the application manager creates an additional algorithm file (a system configuration file *.sys from FBDK), each containing the specific IP and port for the target device. For the communication with the FBDK execution environment, the application manager generates different port numbers for each algorithm execution session.

Several design options were tested and better evaluated using this prototype. At the application manager level we had the possibility of implementing the Publish/Subscribe messaging as a software object, or to have a direct mapping to the corresponding parameters in the algorithm source file. While the second solution was easier to implement, only by use of the communication software objects the system allowed modifying the algorithm parameter values at runtime. This provides greater flexibility as the user can adjust the execution more easily and have a faster process response.

5. Conclusions

This paper describes an approach for the development of a web-based IEC 61499 function block execution system capable of sending the execution results to a remote device. The system integrates different open technologies as software modules, easing the debugging and communication process. The authors adapted the MDA methodology for including different software modules while keeping the generic and easy reconfigurable characteristics. The presented prototype comes to demonstrate the system's opportunities and advantages. This system further develops the programming capabilities of IEC 61499 and opens the possibility for designing other web-based or remote execution applications, like web-based HMI, cloud execution of algorithms, web GPS tracking applications, power systems Smart-GRID integration etc.

Some open issues that need to be addressed in future work refer to the analysis of how the system can support running more instances of the same algorithm, system security, process feedback and IP conflict resolution. All these capabilities must be added before implementing it in a real-time process application.

REFERENCES

[1] *G. Florea, R.Dobrescu, D. Popescu, L. Ocheana, O.Rohat,* PH Center a step to the next generation of Process Control Architecture, Proceedings of the 11th WSEAS International Conference on Systems Theory and Scientific Computation, aug 2011.

[2] *P. Ferreira, V. Reyes, J. Maestre,* A Web-Based Integration Procedure for the Development of Reconfigurable Robotic Work-Cells, International Journal of Advanced Robotic Systems, Jan. 2013.

[3] *R. W. Lewis,* Modelling control systems using IEC 61499: applying function blocks to distributed systems, Control Engineering Series 59, IEE, U.K., 2001.

[4] *V. Vyatkin,* IEC 61499 as Enabler of Distributed and Intelligent Automation: State of the Art Review", IEEE Transactions on Industrial Informatics, vol.7, issue 4, 2011, p. 768 – 781.

[5] *T. Strasser, J. H Christensen, A. Valente, J. Chouinard, E. Carpanzano, A. Valentini, H. Mayer, V. Vyatkin, A. Zoitl,* The IEC 61499 Function Block Standard: Launch and Takeoff, ISA Automation Week 2012.

[6] *L. Noabeb,* XML and PHP Simplified, www.webreference.com.

[7] *D. Flanagan,* Java in a nutshell, Third Edition, O'Reilly press, November 1999.

[8] FBDK, Resources for the New Generation of Automation and Control. Function Block Development Kit (FBDK), www.holobloc.com/.

[9] *V. Vyatkin,* IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design, ISA Publishing, USA, 2012.

[10] *F. Andren, T. Strasser, A. Zoitl, I. Hegny,* A Reconfigurable Communication Gateway for Distributed Embedded Control Systems, Proc. of 38th Annual Conference of the IEEE Industrial Electronics Society 2012, October 25-28, 2012, Montreal, Canada.

[11] *Y. Tipsuwan, M.Y. Chow,* Control methodologies in networked control systems, Control engineering practice 11, 2003.