IEC 61499 Distributed Control Enhanced with Cloud-based Web-Services

Evgenii Demin[1,2], Sandeep Patil[1] , Victor Dubinin[2], Valeriy Vyatkin[1,3]
[1]Luleå University of Technology, Luleå, Sweden
[2]Penza State University, Penza, Russia
[3]Aalto University, Helsinki, Finland
{evgenii.demin@ltu.se, [2]sandeep.s.patil@ieee.org, victor_n_dubinin@yahoo.com, vyatkin@ieee.org }

***Abstract*—This paper presents a framework for employing cloud- deployable web services in the design of distributed control systems in industrial automation. The paper demonstrates how a part of the control logic can be developed as a web service and deployed in the cloud to improve reusability and flexibility. In order to demonstrate the use of the framework we consider an example of Pick-and-Place Manipulator, which was originally designed as IEC 61499 function block application with a distributed control consisting of a high level and a low-level control logic. Firstly the high level control and its functionality is converted to a web services and deployed to a cloud. Secondly the application is modified such that the low level control interacts with these newly developed web services. The paper also presents an interface between low level control and web service using dynamic linked library that bridges communication between the two components It concludes with the simulation results of the of Pick-and-Place Manipulator implemented using the proposed framework.**

Keywords: Web Services, SOA, modular mechatronic systems, Cloud, Industrial Automation, Distributed Systems.

I. INTRODUCTION

The majority of the automated machines' manufacturers and system integrators use modular machine architectures to increase systems flexibility and reduce design effort. At the same time, automation software largely remains monolithic which contradicts with the mechanical structure of machines and hampers systems engineering. The current techniques used by these machine vendors and system integrators in developing the control software for machines in various industrial automation applications are still based on the centralized control.

Improving reusability of existing programs and software efficiency has become an important area of research in industrial automation [1]. Service-oriented architecture (SOA) [2] is a software design pattern based on distinct small modules of software providing application functionality (as services) to other larger applications, which makes it attractive for industrial automation [3-5]. Implementation of SOA is based on a set of open vendor-independent standards, such as web-services [6]. By deploying these services in the cloud, one can also benefit from the many advantages of the cloud computing for making systems more intelligent, re-configurable, flexible and reusable.

In this paper, we propose a new framework for software development in automation systems that is based on SOA and cloud computing in order to achieve flexible and efficient automation systems with decentralized control logic. In this approach, some of the common and computationally heavy jobs are deployed to the cloud and made available as services. The distributed control software modules call these services on a need basis for the overall system operation.

In this paper, the main control logic is implemented using the IEC 61499 standard. The International Electrotechnical Commission (IEC) has developed the IEC 61499 standard [7-10] for design of distributed control systems. It is the successor to the IEC 61131-3 standard that focused on the design of control systems based on programmable logic controllers (PLCs). The IEC 61499 focuses specifically on the development of distributed automation systems. It promotes such features as modularity, reusability, flexibility, extendibility, interoperability, portability [11] and re-configurability. The standard is enabler of open, hardware independent software development frameworks for distributed automation systems [8], in particular capable of system-level simulation and verification of distributed control software. The standard has been used actively in many research areas [8] including smart grids [12, 13], IEC 61499 has been most recently used in the SOA based applications [14-17] and this paper builds on top of these contributions.

We propose to use web services with following advantages in mind.

- The PLC's have a fixed scan cycle, any computation that takes more time than the scan cycle time, will result in out of sync issues and undesired behavior. Hence any software piece of code with such long computation can be converted to a web service.
- By having such a web service, upgrading/changing the piece of software becomes very easy, as it does not involve redeploying the control software to the PLC's and hence maintenance downtime is reduced
- Highly scalable distributed control systems can be developed using IEC 61499 based applications that interact with these web services.

The rest of this paper is structured as follows. Section 2 describes the case study example in detail. Section 3 details

978-1-4799-8389-6/15/$31.00 ©2015 IEEE

about the web service implementation. Section 4 presents the integration of function block application with the web service. Application simulation and working mechanism is presented in Section 5. Section 6 proposes use of web services in cloud computing and finally Section 7 concludes the work.

II. Model of Pick-And-Place Manipulator

This study was performed with the use of an existing IEC 61499 application for a modular pick-and-place (PnP) manipulator model that is described in [18, 19]. The PnP manipulator consists of a system of pneumatic cylinders and suction unit that moved work pieces from input trays to an output tray (Fig. 1(b)). The control logic of the manipulator and scheduling of its operations were implemented in a fully decentralized way to support the plug and play integration of intelligent mechatronic components [20-22].

The function block application for the PnP manipulator was developed as an automated system composed from intelligent mechatronic components, pneumatic cylinders in this case. Multiple configurations of the manipulator have been studied in the previous publications [18, 19]. But in this work we used a configuration with 6 cylinders (3 horizontal and 3 vertical) and a suction unit. Each cylinder can extend or retract, driven by the corresponding control signals emitted by its embedded controller. It is assumed that information on particular properties of a cylinder (such as length, speed, etc.) is well known to the cylinder's embedded controller, but not to other cylinders.

The decentralized scheduling scheme proposed in [23] is used to schedule the operation of the PnP manipulator. That is, the distributed scheduler determines at runtime the combination of cylinders that will participate in the job. The manipulator's operation is divided into two stages: planning and execution. During the planning phase, the system determines which cylinders need to be extended in order to reach the required position.

The manipulator's control logic is modelled and implemented using the IEC 61499 international standard for distributed automation in the NxtStudio 2.0 [24] development environment. NxtStudio is a commercial software used in the design and deployment of IEC 61499 based applications. Fig. 1 shows the six cylinder pick-n-place manipulator with the distributed control (Fig. 1 (a)) and it can be seen that each cylinder control has the scheduling part (planning) and actuation part (execution) (Fig. 1 (c)). The "*CylSchd*" function block is responsible for planning and "*CYLACT*" function block is responsible for execution part.

The decentralized control scheme enables plug and play design of such machines from available mechatronic components, including their extension and substitution of parts in a vendor-independent way, which may have substantial business benefits in terms of flexibility and maintainability of manufacturing plants.

Even more flexibility could be achieved if we used services located in the cloud. This way services can be updated easily and extended during the lifecycle of the mechatronic components. In this paper we investigate such a possibility by implementing such services as web services that interact with each other. The web services perform functions for scheduling of which cylinders participate in a particular job of picking a workpiece from a particular source position and dropping it at a destination. In Fig. 1 (c), we convert the scheduling part (logic inside "*CylSchd*" function block) to a service and covert the "*CylSchd*" from a basic function block to a Service Interface Function Block (SIFB) that interacts with the newly created web service.

III. Implementation of Cylinders Scheduling Web Service for PnP Manipulator

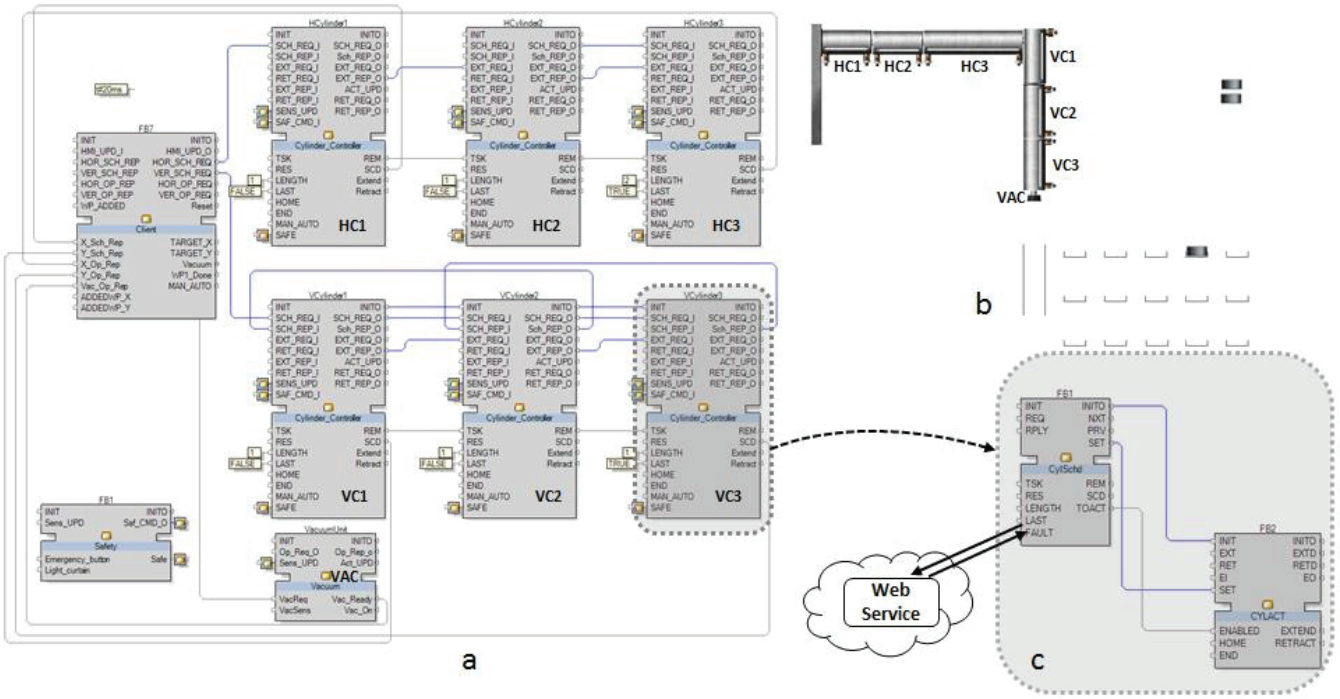

Fig. 1. Pick-and-Place manipulator and its decentralized control modelled in IEC 61499.

In this part, we describe how to convert the scheduling part to services using service-oriented architecture and demonstrate its advantages.

In the service-oriented architecture, a web service communicates and interacts with its clients using SOAP messages typically using the HTTP protocol. To call a web service, one needs to know the corresponding URL, as well as the name of the web service and a list of its input parameters. A web service can serve an unlimited number of clients.

In the context of our example, the development of a web service does not impose any restrictions on the number of cylinders in the automated system, as well as their minimum and maximum size. The cylinders may be of different length but in our work we assume that their length must be a multiple of the space between the two sources of workpieces. The algorithm is designed in such a way that this web service performs search of all possible ways to perform the required task.

This imposes some overheads in computing, but extends the capabilities of the algorithm. This allows one to perform an optimization of the system of cylinders and choose the most suitable option for performing the job, an example of such optimization could be selection of a combination that results in the lowest energy consumption. The web service also supports for transmitting information about the cylinders, say for some reason, a cylinder can't make delivery of the items (example: require replacement or maintenance or there is a system fault), then such information is stored for corresponding cylinder. In this case, the web service will perform planning, taking into account information about the received faults. Below is the code snippet for the schedule service.

```
public class ShedulePnP //Schedule PnP
Service {
     ... static List[17] Schedule(int
     Coordinate,
     List<int> cylLength,
     List<int> brokenCylinders)
     {
       ...
     //returns an array with the values
     TRUE/FALSE and reminder of the job (X
     or Y values)     }
}
```

The *ShedulePnP* class contains a function *Schedule* that is implemented for cylinder scheduling. Scheduling is carried out separately for horizontal cylinders and vertical ones. Function's input parameters are coordinates of the location of the workpiece source, and an array containing the values of the lengths of the cylinders that will participate in scheduling and an array containing details about faulty cylinders.

The developed algorithm allows to perform scheduling in parallel (schedule of horizontal cylinders can be run in parallel to schedule of vertical cylinders). Moreover, the scheduling algorithm itself can be implemented as several parallel operating services, each of which corresponds to a particular phase of the algorithm. A more detailed description and application of this approach can be found below in section IV.

IV. Integration of Web Services with Function Blocks

In order to use web services inside a function block application a software interface for data transfer via HTTP protocol is required. There is no such interface implemented in the standard libraries of function blocks provided by *NxtStudio*, therefore we had to integrate existing libraries with the function block application.

In IEC 61499 the applications' functionality is programmed in basic function blocks, in which algorithms are programmed in a variety of languages supported by a particular development environment. In NxtStudio only Structured Text language is supported and is quite restricted. In order to develop interfaces there are two options available: Service Interface Function Blocks (SIFBs), which are programmed in C, and Composite Automation Types (CAT), programmed in C# and specifically suitable for development of human-machine interface functionalities. Ideal option will be to use SIFB's, but for sake of convenience we chose the CAT mechanism for this paper, because it allows us to use the standard .NET libraries for web services, instead of writing new library in C (used in SIFB's).

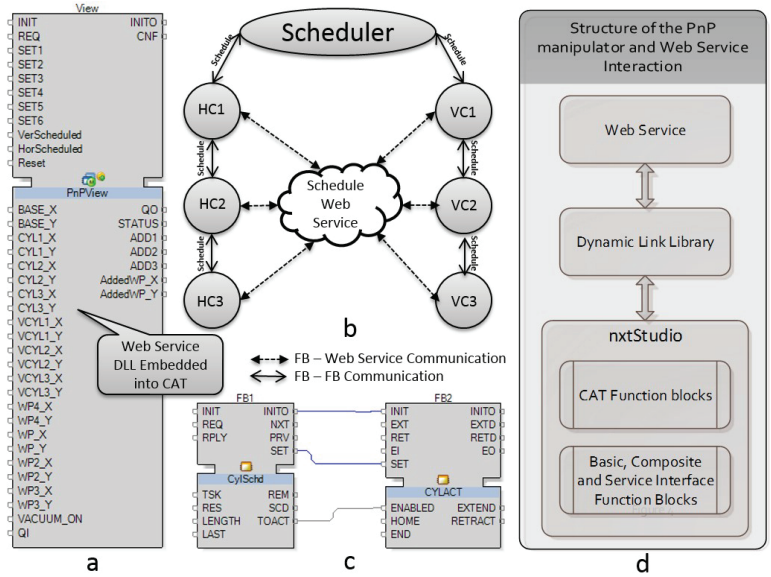

Fig. 2. (a) CAT function block with embedded DLL to interact with Web-Services, (b)the control flow in the IEC 61499 Distributed control, (c) "*CylSchd*" now interacts with the CAT block as in (a) and (d) Overview of interaction between Function Block application and Web Services.

A CAT-block is also a function block type that combines HMI and also control logic in the form of traditional IEC 61499 composite function blocks. CAT-block allows us to connect an external DLL-library and provide a channel to the necessary functions. Fig. 2 shows the block diagram of communications and implementation details of the FB application interaction to the web services. Fig. 2(b)

Fig. 2(a) shows the CAT block with integrated DLL for web service interaction. CAT function block transmits the coordinate's details of the work piece location to the web service. After the scheduling is done by the web service, the CAT block retrieves output data (the result of the call is an array containing the list of cylinders that will participate in the given task) and performs the pick and place operation of the workpiece in accordance with the received Schedule.

The scheduling of the horizontal and vertical cylinders is done separately as shown in Fig. 2(b). In order to perform this, we need to call a web service for each. An example of a call is given below.

```
...
Uri uri = new
Uri("http://localhost:1577/ws1.asmx");
WebServiceInvoker ws = new
WebServiceInvoker(uri);
string res = ws.LoadService(uri, "Service1",
"ShedulePnP", "225:300.75.150.75:"); res +=
ws.LoadService(uri, "Service1",
"ShedulePnP", "150:75.75.150.75:2");
...
```

The *LoadService* function, which is compiled as a DLL, calls the web service *ShedulePnP*. All input and output parameters are variables of the string type, which increases clarity and helps for scaling of the project code. All conversions from string values to numeric and back are performed in the DLL library. The value of the coordinates of the workpiece, the array of length of the cylinders, and the array with the numbers of idle (or faulty) cylinders are transmitted as a single string value. The symbol ":" serves as a separator of these values, the character "." acts as the separator of the array elements.

The format of the string is as shown below.

```
<coordinate>:<length of Cylinder1>.<length of
Cylinder2>…<length of CylinderN>:<broken
Cylinder1>.<broken Cylinder2>…<broken
CylinderN>.
```

Fig. 2(d) summarizes the overall Function Block application and web service interaction.

V. Simulation of Pick-and-Place Manipulator

The proposed method was applied for the control and operation of a simulated PnP manipulator. For our case study we assume the mechanical structure of the manipulator is fixed (with six cylinders), but the operation can be interactively tested on any possible configuration of input trays.

The simulation setup is shown in Fig. 3, the left hand side part of which contains animated model of the PnP and combination of input trays in the right bottom. A work piece can appear in any the 20 positions (4 rows x 5 columns) and it needs to be picked up by PnP manipulator and dropped to the output tray. However, not all positions of input trays may be reachable by the manipulator in its particular mechanical structure. This shall be established during the scheduling phase. The work piece selection panel allows interactive clicking on the input tray). It also displays the manipulator operation, fault/unavailability of cylinders (crosses on cylinders in the HMI), along with a hand symbol (representing light curtain sensor) and an emergency stop button that can be pressed to simulate the real manipulator operation condition (Providing EP).

The last row and column of this tray matrix are deliberately made not reachable by the current length and number of cylinders; however, adding more cylinders can make them accessible. The remaining trays' reachability will depend on the instant availability of appropriate cylinders in each horizontal and vertical line (if they are not faulty). Now, once an input tray is selected, if it is accessible by the available horizontal and vertical cylinders

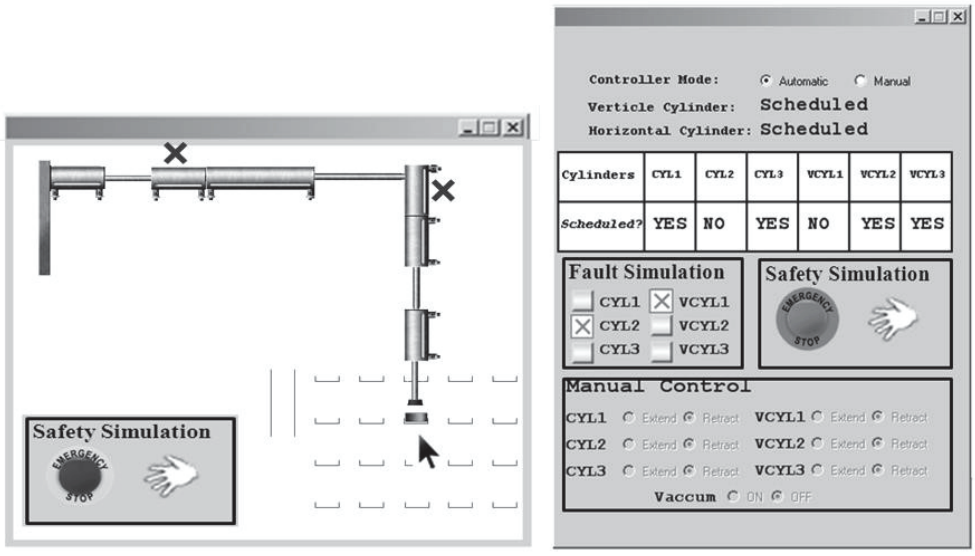

Fig. 3. Visualisation of the simulated PnP operation with interactive input generator (left hand side), and scheduling results display with interactive fault generator (right side).

(which is established collaboratively by cylinders' web services during the scheduling phase), the manipulator starts operating immediately; otherwise, the manipulator will stand still until the next reachable WP tray is selected.

If there is a physical intervention (such as operators' hand comes in the way and detected by light curtain senor) or emergency stop button is pressed, the operation will be suspended until the WP tray is selected again (to indicate safe condition) in order to resume its operation.

The right hand side panel is aimed for selection of the operation mode (automatic or manual), to be able to test the cylinders manually and monitor the scheduling status of the manipulator operation under faulty and unsafe conditions. A fault in the cylinder operation is simulated by a number of push buttons with which each cylinder can be disabled or enabled. Also the emergency stop situation can be simulated through two push buttons.

VI. Implementation Of Services in the Cloud

There are lot of areas where cloud computing is used and industrial automation systems are not an exception. Existing systems can be easily extended to a full-fledged cloud computing architecture. On the basis of the framework presented in this paper, we can suggest the use of multiple "competing" web services such as the one used in [3]. With the help of several web services, the scheduling can be parallelized and the overall load on the system can be reduced. In our example case study this was demonstrated by doing in parallel the scheduling of horizontal and vertical cylinders.

Web services may be hosted at servers located at any geographical location. A group of web services can significantly improve fault tolerance and speed of computation. Web services are capable of the duplex exchange of information. The number of web services can be increased or decreased depending on the system requirements. With this an automated system is easily scalable and reconfigurable [4].

From the application point of view, one can use a repository (a database files) that keeps track of services in the cloud. The repository needs to store information about available web services and contains the functions of registration, deletion, and a list of available services for a given task. Through the use of the repository, one can switch on or off these web services in real-time without stopping the current process. This way the system allows you to use an unlimited number of web services, with information about each web service stored in the repository.

When a new web service is added to the pool, web services perform registration in the repository and send service information for further interaction. Storing information in the repository can be implemented using a database or a file. The repository can also perform the function of distribution and load balancing.

Structure of the organization of the simulation and control model of PnP manipulator on the basis of cloud-based web services is presented in Fig. 4.

Such an approach may find application in industrial automation applications that require high performance, fault tolerance, scalability and portability.

VII. Conclusion

The paper demonstrated the use of cloud deployable web services for industrial automation systems. It was shown that the use of service-oriented architectures for systems with cloud computing and distributed data processing works well. The proposed methodology was implemented on a small pick-n-place manipulator using NxtStudio 2.0 and Visual Studio 2010 for DLL that interfaced between the function block applications and the web services. The example used here is a proof of concept for application in real control systems for industrial automation. The advantages of using such systems in production are obvious: high processing speed, high degree of integration and scalability, as well as ease of maintenance.

Some limitations of the cloud deployment of services may be in terms of performance, but, there could be performance gains, in case when all services conducting the scheduling (in our case) are located in the cloud, so that the schedule search is performed there with a more capable hardware platform than microcontrollers (or PLC's) that are embedded into cylinders.

Model of PnP manipulator was chosen to illustrate the benefits of sharing the functional units and Web services in the design of distributed automation systems.

Future work will include implementation of the developed framework on the device level, along with investigation of verification and validation means for the Cloud/enabled function block applications, including formal modelling frameworks, such as [25].

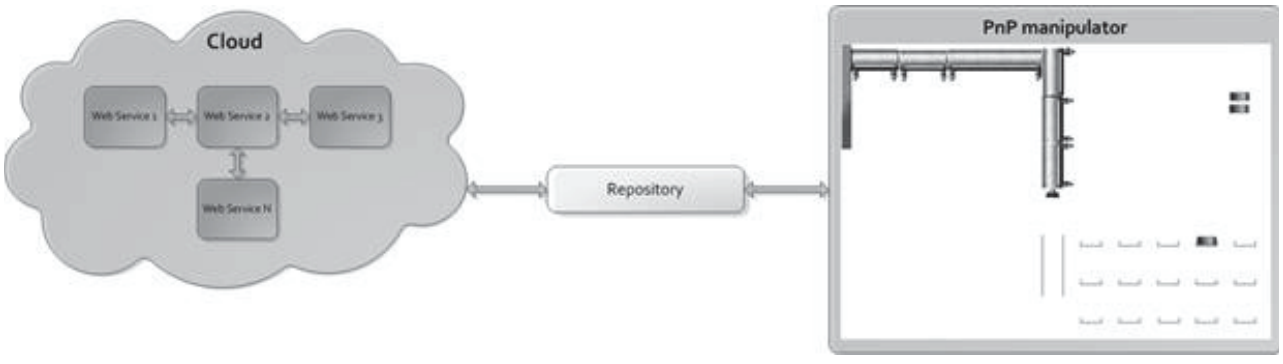

Fig. 4. Using the cloud-based web services in PnP manipulator.

VIII. ACKNOWLEDGMENTS

This work was partially supported by Luleå Tekniska Universitet, grants 381119, 381940 and 381121 and by the Federal Targeted Programme "Research and development in priority areas of elaboration of STC of Russia for 2014-2020" (agreement of 19.06.2014 № 14.574.21.0045).

IX. REFERENCES

1. Vyatkin, V.: Software Engineering in Industrial Automation: State-of-the-Art Review. Industrial Informatics, IEEE Transactions on 9, 1234-1249 (2013)
2. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR (2005)
3. Delamer, I.M., Lastra, J.L.M.: Loosely-coupled Automation Systems using Device-level SOA. In: Industrial Informatics, 2007 5th IEEE International Conference on, pp. 743-748. (Year)
4. Jammes, F., Smit, H.: Service-oriented paradigms in industrial automation. Industrial Informatics, IEEE Transactions on 1, 62-70 (2005)
5. Lobov, A., Lopez, F.U., Herrera, V.V., Puttonen, J., Lastra, J.L.M.: Semantic Web Services framework for manufacturing industries. In: Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on, pp. 2104-2108. (Year)
6. Richardson, L., Ruby, S.: Restful web services. O'Reilly (2007)
7. Function blocks — Part 1: Architecture, IEC Standard 61499-1. (2012)
8. Vyatkin, V.: IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review. IEEE Transactions on Industrial Informatics 7, 768-781 (2011)
9. Vyatkin, V.: IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design. ISA (2007)
10. Lewis, R., Engineers, I.o.E.: Modelling Distributed Control Systems Using IEC 61499: Applying Function Blocks to Distributed Systems. Institution of Engineering and Technology (2001)
11. Patil, S., Yan, J., Vyatkin, V., Cheng, P.: On composition of mechatronic components enabled by interoperability and portability provisions of IEC 61499: A case study. In: Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on, pp. 1-4. (Year)
12. Patil, S., Vyatkin, V., McMillin, B.: Implementation of FREEDM Smart Grid distributed load balancing using IEC 61499 function blocks. In: Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE, pp. 8154-8159. (Year)
13. Zhabelova, G., Patil, S., Chen-Wei, Y., Vyatkin, V.: Smart Grid applications with IEC 61499 reference architecture. In: Industrial Informatics (INDIN), 2013 11th IEEE International Conference on, pp. 458-463. (Year)
14. Dai, W., Vyatkin, V., Christensen, J.H.: The application of service-oriented architectures in distributed automation systems. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 252-257. (Year)
15. Dai, W., Riliskis, L., Vyatkin, V., Osipov, E., Delsing, J.: A Configurable Cloud-Based Testing Infrastructure for Interoperable Distributed Automation Systems. In: IEEE International Conference on Industrial Electronics IECON 2014. (Year)
16. Dai, W., Christensen, J.H., Vyatkin, V., Dubinin, V.: Function block implementation of service oriented architecture: Case study. In: Industrial Informatics (INDIN), 2014 12th IEEE International Conference on, pp. 112-117. (Year)
17. Chia-Han, Y., Vyatkin, V., Cheng, P.: Model-Driven Development of Control Software for Distributed Automation: A Survey and an Approach. Systems, Man, and Cybernetics: Systems, IEEE Transactions on 44, 292-305 (2014)
18. Sorouri, M., Vyatkin, V., Salcic, Z.: Rule-based composition of intelligent mechatronic components in manufacturing systems using prolog. In: Industrial Informatics (INDIN), 2013 11th IEEE International Conference on, pp. 242-247. (Year)
19. Sorouri, M., Patil, S., Vyatkin, V.: Distributed control patterns for intelligent mechatronic systems. In: Industrial Informatics (INDIN), 2012 10th IEEE International Conference on, pp. 259-264. (Year)
20. Vyatkin, V., Hanisch, H.-M., Pang, C., Yang, C.-H.: Closed-loop modeling in future automation system engineering and validation. IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews 39, 17-28 (2009)
21. Cheng, P., Vyatkin, V.: IEC 61499 function block implementation of Intelligent Mechatronic Component. In: Industrial Informatics (INDIN), 2010 8th IEEE International Conference on, pp. 1124-1129. (Year)
22. Vyatkin, V.: Intelligent mechatronic components: control system engineering using an open distributed architecture. In: Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference, pp. 277-284 vol.272. (Year)
23. Pang, C., Vyatkin, V.: Systematic closed-loop modelling in IEC 61499 function blocks A case study. In: IFAC Workshop Series. Elsevier Ltd. Books Division, (Year)
24. www.nxtcontrol.com
25. Sorouri, M., Patil, S., Vyatkin, V., Salcic, Z.: Software Composition and Distributed Operation Scheduling in Modular Automated Machines. submitted to an IEEE journal, under review