A Case Study on Migration from IEC 61131 PLC to IEC 61499 Function Block Control

William Wenbin Dai, wdai005@aucklanduni.ac.nz
Valeriy Vyatkin, v.vyatkin@auckland.ac.nz
Department of Electrical and Computer Engineering
University of Auckland, Auckland, New Zealand

Abstract - **The IEC 61499 architecture is a new standard proposed to replace current PLC technologies. This paper aims to develop a guide to migrate from IEC 61131 PLC technologies to IEC 61499 function blocks, especially for PLC programmers. Multiple PLC platforms are chosen as examples and transformed function block system is provided. Beyond migration rules, current limitation and cautions during migration are discussed on the final section of this paper.**

I. INTRODUCTION

The IEC-61131[[1]] standard for Programmable Logic Controller (PLC) programming has been widely adopted by vendors and users of automation technologies. However, there is growing understanding of the IEC 61131-1 limitations for achieving flexibility and reconfigurability of automation systems. Thus, although all major PLC vendors are designing their products according to IEC-61131 standard, some incompatibility between different PLC brands is still the case, which limits flexibility of substituting one brand PLC by another. The need for decentralized control is ever growing for many reasons, such as flexibility and reliability (a centrally controlled system implies single point of failure in case of a break down in any part of the system), but PLCs are not good for implementing distributed control.

The International Electrotechnical Commission (IEC) has developed a new standard – IEC 61499 [[2]] to make automation systems reconfigurable, interoperable and portable. IEC 61499 is based on the concept of function blocks. The IEC 61499 type function block is event-driven which differs from IEC 61131 type function block. IEC 61499 is in early ages of its adoption by industry, but there is a huge potential of growth in future.

Nowadays, modularization and reusability are taking the ever increasing importance in PLC programming. Especially for medium and small size firms, modularized code promises more efficiency and less re-work time. The productivity will be improved significantly by deploying IEC 61499 function blocks. Small companies will be able to build their own intellectual properties in IEC 61499 function blocks into a components library and allow re-use in future.

The problem of migration from IEC 61131-3 to IEC 61499 has recently attracted attention of researchers, e.g. works [[3],[4]]. In this paper we continue investigation of this issue from a practical perspective of material handling systems automation. Material handling systems has been considered as a perfect application area for the new IEC 61499 architecture due to the high modularity of the machinery. In the works [[5], [6]] the corresponding function block architecture for MHS was proposed and investigated. A migration path from one standard to another could save lots of efforts on adoption of the new standard thus making it more attractive to industry. At this stage we address the problem at rather informal level, more caring about feasibility of the migration rather than subtle details.

The paper is structured as follows. Fundamental information of both IEC 61131 Function Block Diagram and IEC 61499 Function Block are provided in Section 2. Then a case study of a conveyor system will be introduced. Both IEC 61131 and IEC 61499 implementations are presented in Section 3. And finally the translation rules summarized from case study are discussed in Section 4.

II. COMPARISON BETWEEN IEC 61131 FUNCTION BLOCKS AND IEC 61499 FUNCTION BLOCKS

A. *IEC 61131 Function Blocks*

As defined in IEC 61131-3, standard programming languages for PLC are Structured Text (ST), Ladder Diagram (LD), Instruction List (IL), Sequential Function Chart (SFC) and Function Block Diagram (FBD). PLC vendors implement PLC programming languages according to the IEC 61131-3 standard, but not limited to it. This causes serious compatibility. For instance, despite similar look, a ladder diagram from one vendor is impossible to import to another ladder diagram designed by another vendor.

The IEC 61131-3 standard's function block (in FBD) is a subroutine with parameters and local data. However, syntax of particular implementations may include many vendor-specific details. Inside a function block, not all standard IEC 61131 programming languages are available, e.g. SFC is not supported in Allen-Bradley

Add-on instructions (Rockwell Function Block Implementation). Also the scope of access to controller memory is different between vendors. Some PLCs only support memory/variable/tags associated within the block (Local tag/memory) but others can access global variables.

B. IEC 61499 Function Blocks

IEC 61499, introduced by IEC in 2005, is considered as a next generation enhancement of PLC system engineering. The basic principle of IEC 61499 is event-driven function blocks (FB), which are invoked only when an event arrives to one of their event inputs. The same time some specified data inputs will be updated. During rest of the operation time the FB remains idle. This will significantly improve the efficiency and reduce computing power consumption and communication bandwidth. Also, using FBs as the top-level representation provides a complete system overview with all devices, communication layouts and programs. The function blocks projects can be easily ported to other machines. Encapsulation of functions into function blocks increases their reusability. As a result, standard component library of function blocks shall provide maximum efficiency for saving project re-work time.

For the details of IEC 61499 we refer the reader to [7]. Here we provide only most essential terminology.

There are three types of function blocks: basic, composite and service interface. A basic function block is the fundamental element in IEC 61499 function block system. A basic function block must contain an Execution Control Chart (ECC) – a state machine with conditional branches and corresponding algorithms executed in states. IEC 61499 Function Blocks have two types of inputs and outputs: events and data. A Function Block will be only executed when an input event is triggered. Also data inputs and outputs associated with that particular input will be updated (association is shown as a vertical line connecting an event input/output with data inputs/outputs). As defined in IEC 61499, the internal algorithms of function block can be written in a number of languages, such as IEC 61131 programming languages (ST, LD, IL, SFC and FBD), and high level programming languages like C or Java.

A network of basic and composite function blocks forms body of a *composite* function block. This way hierarchical structure can be built, increasing reusability of code. Similar to basic FBs, the IEC 61499 composite function block has its own interface. The inputs and outputs of a composite function block can be connected directly to inputs (outputs) of the component FBs.

Finally, a network of basic and composite FBs forms an application. System configuration combines the application logic with device topology, abstract definition of communication networks and exact mapping of function blocks to devices.

Service Interface Function Blocks (SIFB) are destined for wrapping hardware dependencies of applications. SIFB is considered as a ‘Black Box’. The forms of definition of SIFBs internal logic are not very restricted by the standard. A SIFB is defined by a number of event sequence specifications, describing interaction between resource (hardware) and the FB. This way of specification can be useful for documentation of SIFB functionality, especially if it is hidden or written in a low-level programming language. SIFBs can be used to implement various communication protocols, interfaces to databases or human-machine interfaces (HMI).

C. Programming Tools

There are several IEC 61499 programming tools available, for example: FBDK[[8]], ISaGRAF[[9], [1], and FBench [[10]].

FBDK is the world first IEC 61499 programming tool, it is written in Java and implements function blocks as Java classes. FBDK is widely used in academia and in research community. ISaGRAF is the first commercial PLC programming tool supporting IEC 61499 standard, with a few thousands of installations of the last version (v.5) supporting IEC 61499. FBench is an open source project initiated by OOONEIDA [[11]]. FBench is capable for IEC 61499 FB design, development, debugging, run IEC 61499 application, verification etc. FBench open-source project is continuing by our research group at the University of Auckland. FBench aims to be a complete programming tool set, which supports IEC 61499 and IEC 61131-3 programming languages.

As defined in IEC 61499, any high level programming languages such as Java or C can be used for writing internal FB logic. FBench is fully compatible with IEC 61499 standard and convenient to target to any IEC 61499 controller or runtime supporting IEC 61499. In this example, FBench is selected as the demo programming tool. All IEC 61499 software currently available are more for evaluation propose rather than for commercial use which means they are lack of support and not compatible with most PLC controllers.

For this case study, Siemens Step 7 with S7-300 PLC is selected as PLC demo example and a TCS-NZ MO’intelligence embedded controller is used to deploy IEC 61499 function blocks.

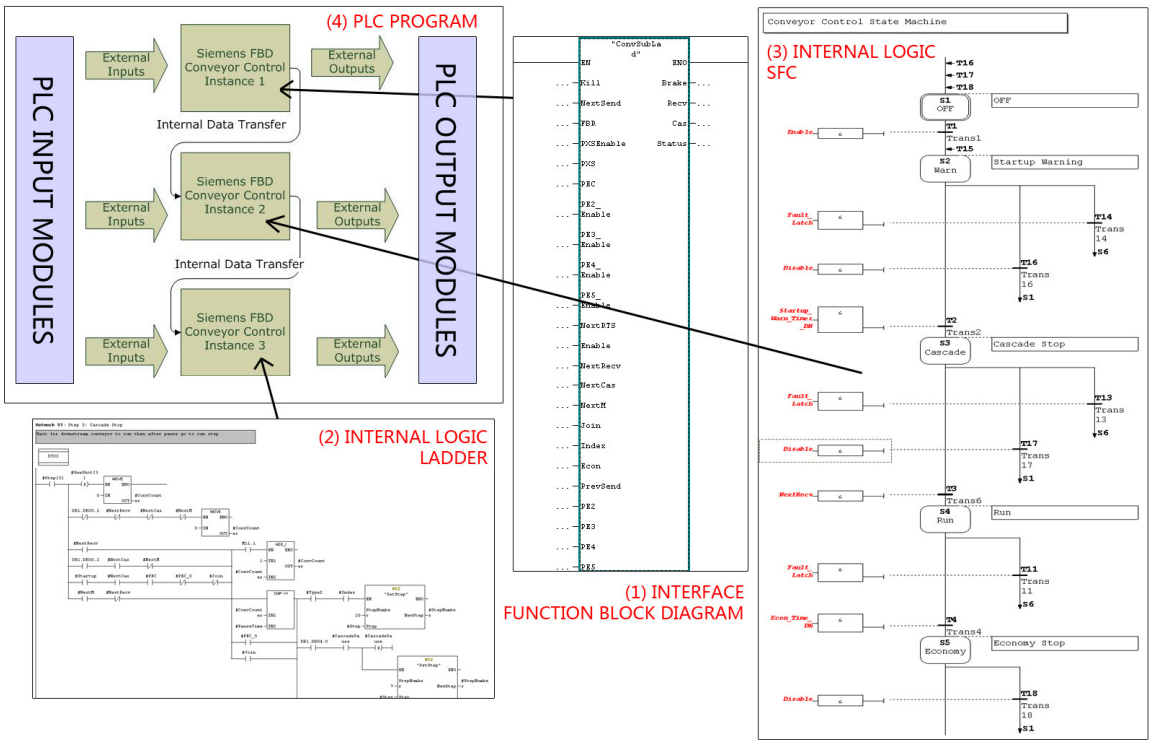

Fig.4. PLC Implementation of Control Conveyor System

III. Conveyor Control Case Study

In this paper, a conveyor control example will be used to illustrate features of new IEC 61499 function blocks.

A. Basic Conveyor Control Concepts and Sample System Overview

The major components of a general transport conveyor are conveyor belt, variable frequency drive and photo eye. To control the conveyor, a series of feedback inputs and command outputs are required. Also several status indication signals of the conveyor are produced. Normally conveyor can run either in auto mode or manual mode. A control panel consists of switches and buttons to allow full control of a particular conveyor.

The test bed chosen in this paper is a series of general transport conveyors, as described in Figure 2.

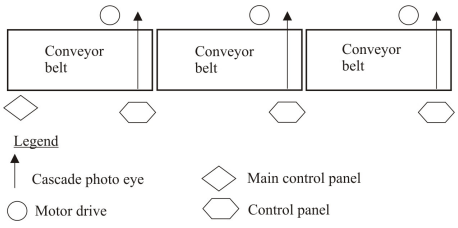

Fig.2. Sample Conveyor Test Bed

Conveyors are linked together head to tail. Each conveyor has an auto/manual control panel attached to the side. A sounder and beacon for alarm purposes are installed to indicate warnings or fault alarms.

Control of a single conveyor is initially designed as a state machine (Figure 3). A general transport conveyor state machine consists of Off, Startup, Run, Economy stop, Cascade stop and fault states. Each state has its own algorithms associated.

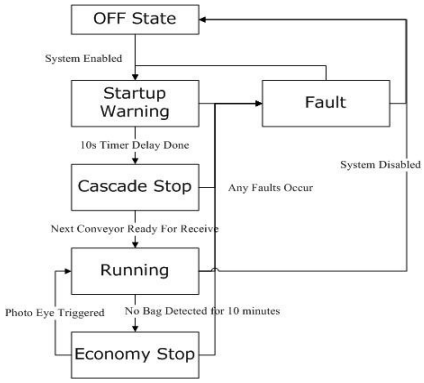

Fig.3. Conveyor Control State Machine

Meaning of the states is as follows. The ‘Off’ state is the initial state of conveyor control routine. As soon as the test bed’s operation is enabled, the current state shall jump to startup and generate warning. Once this warning period is complete, the most downstream conveyor will start running, followed by upstream ones. When a bag reaches the end of the conveyor chain, the last conveyor in the chain shall stop immediately. Rest of belts in the system will perform cascade stopping when bags hit each cascade electrical photo eye located at the tail of

those conveyors. Also when there is no bag detected for a period of time, for example, 30 seconds, the conveyor shall turn into economy mode to save power. When a conveyor is in any state except the 'OFF' state, and a fault occurs, the conveyor will turn into the 'Fault' state. This faulted conveyor shall remain in the 'Fault' state until the physical fault is cleared and reset button is pressed.

B. Implementation of conveyor control

1) Siemens PLC Platform

Control of the test bed has been implemented in Siemens Step 7 and simulated on S7-300 PLC. Out of the five programming languages defined in IEC 61131 and supported by Siemens Step 7, the Function Block Diagram is the best top level interface in Siemens implementation for the conveyor control function. As seen in Figure 4, the PLC program structure is described in (4). SFC ((3) in Figure 4) is used inside the conveyor control function block instances to represent the control flow chart. Each state in SFC corresponds to a state in the conveyor control state machine. Transitions between SFC steps are programmed in ladder logic as shown in Figure 4, (2). Siemens Function block interface ((1) in Figure 4) uses input EN to trigger the execution of this function block and output ENO will be turned on after the execution concludes. Before executing an algorithm inside the block, all inputs status shall load into block internal registers. Other logical statements inside a function block, for instance, indicator status, alarm status and emergency stops, are written in IEC 61131 ladder diagram language. All ladder elements are driven by the conveyor control state machine, only relative parts of the ladder logic code according to current state will be executed. In ladder logic, one-shot output is simply not allowed to appear multiple times in entire PLC code. If one-shot output is turned on or off several times in the PLC code, this one-shot output only refers to last appearance, all previous logics will be ignored. In order to solve this issue, in case of multiple states setting a one-shot output the corresponding ladder logic statements will be grouped together.

2) IEC 61499 FB Implementation of conveyor control

The methodology of designing this conveyor control system in IEC 61499 is completely different from the PLC implementation. The top level structure of an IEC 61499 project is system configuration that consists of several devices populated by function blocks networks. On the bottom level of the system hierarchy, each conveyor is represented by a basic function block. The publisher and subscriber communication function blocks are used to exchange data between control FBs and HMI FBs. This communication is based on UDP/IP over Ethernet.

a) Conveyor Control FB

The conveyor control is designed in basic function block.

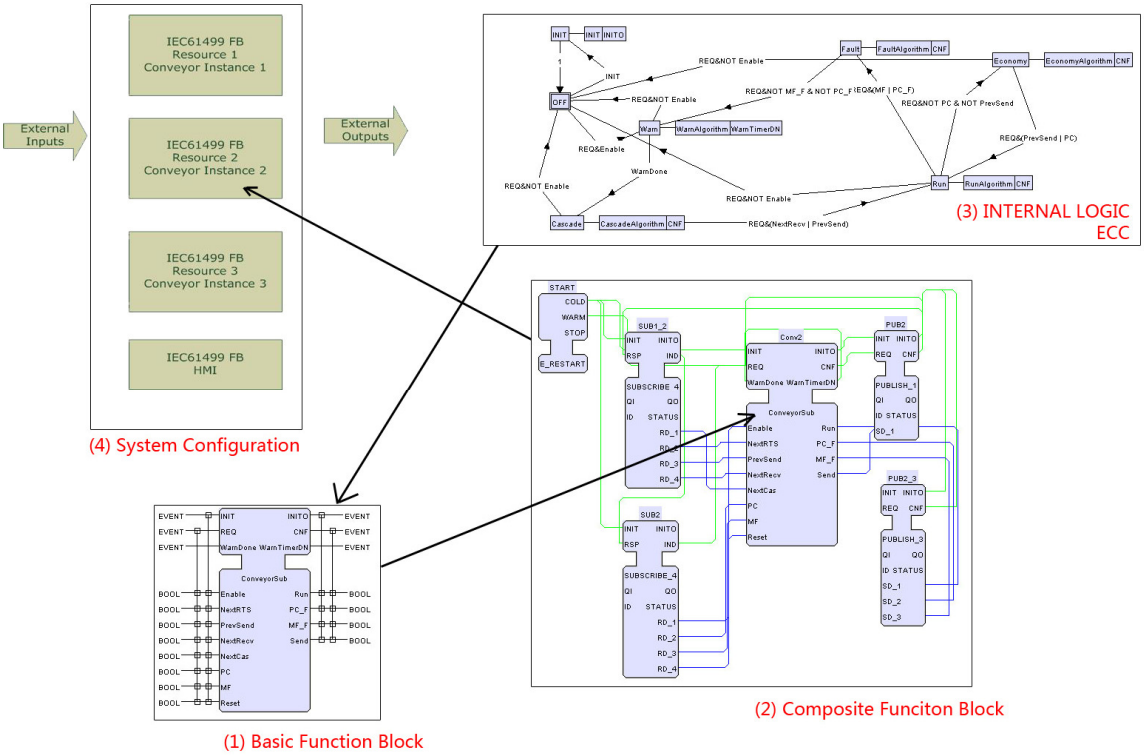

Fig.5. IEC 61499 FB implementation of Control Conveyor System.

Instead of SFC or ladder logic, Execution Control Chart (ECC) is used as internal state machine.

As shown in Figure 5 (3), each execution control state (EC State) corresponds to a single step in PLC program or single state in state machine. The algorithms in each EC State are written in Java. When the input event is triggered, the associated data inputs to that particular event will be read into FB and ECC shall change its state depending on updated event and data inputs. The output events and data also would be updated once the algorithm finishes execution.

b) Human Machine Interface

As one of the significant improvements of IEC 61499 function blocks compared to IEC 61131, the Human Machine Interface (HMI) can be integrated within the function block networks. Within an IEC61499 function block, not just only control algorithms, but also data and HMI are included. Any relevant information of this physical device can be stored inside the block.

For the example below, a main control panel and number of control stations for individual conveyors are built into the function block network to allow clients control the test bed through HMI created by IEC 61499 runtime. In addition, the graphical demonstration of test bed is generated by HMI function blocks and the system can be simulated using FBench. This provides a great convenience for both simulation and production to the end users.

c) System overview

The final step of design an IEC 61499 system is to deploy all basic, composite, HMI and communication function blocks into control devices.

It is a significant improvement of IEC 61499 that the application can be easily subdivided into several distributed devices. Each device is running independently, and exchange information with other devices by using communication FBs. This feature offers improved reconfigurability, redundancy and distribution as compared to the current PLC world. If a part of the system is faulty, for instance, a device is in deadlock, the IEC 61499 system could diagnose faults automatically and reconfigure system in real-time operation.

IV. TRANSLATION RULES AND LIMITATIONS

A. *Summary of Translation Rules*

We summarized a list of translation rules using which programs in IEC 61131 programming languages can be converted into IEC 61499 function blocks.

(1) The IEC 61131 function block diagram (FBD) can be placed into an algorithm of IEC 61499 basic function block with minor changes. One important note is that when a state change depending on a data input, there must be an input event associated with that data input. As the IEC 61499 function blocks are event-driven, without that additional event input, the ECC inside basic function will not be executed.

In most implementations of the PLC vendors, nested function block structures are feasible. To map that, FBD inside an FB shall transfer to composite function block.

Additionally, complex calculations or looping can be re-written in high level programming languages like C or Java rather than ladder logic to simplify the development.

Here is the basic translation rule summary.

IEC 61131 Languages	IEC 61499 Format
FBD	Basic/Composite FB
LD	Basic FB
ST	Basic FB
IL	Basic FB
SFC	ECC*

Fig.6. Translation Rules Table for Programming Languages

*For SFC to ECC discussion please refer to 4.2 and (2) below

(2) The primary premise for converting to ECC from SFC is that no concurrent state exists in the SFC. As defined in the IEC 61499 standard, only one EC state can be in active state at one time. Each EC State in ECC can map to a transition state in a non-concurrent SFC. As ECC only exists in basic function block, so when transfer SFC to ECC, the SFC must have no references to composite FB constructs.

(3) As there is no global variable in IEC 61499, the global data can be encapsulated in one function block and that block needs to be explicitly connected to all function blocks in the application which may need to access the data. This SIFB shall provide read and write access to the memory where the global variables are stored. Also this SIFB must take the role as arbiter to ensure always only one function block can update values at one time.

(4) To reduce the power consumption of entire system, only the necessary algorithms shall be executed in each unit of time. This design will minimize the number of inputs and outputs to be energized in each execution time. As for large scale of system, to prevent unnecessary power consumption is critical for system performance and economy considerations.

(5) In IEC 61131-3, all programs are cyclic tasks, even if it's not necessary to run sequentially and periodically for all tasks. Cyclic task can be also implemented in IEC 61499 as follows: a cycle generating block is required as initial event trigger to provide a cyclical event. Once the last function block in the chain finishes its execution, a "Done" bit is required to pass back to this cyclical event generator to continue. But scan time in PLC is critical for all real-time applications. To prevent a long scan time per cycle, the application must be redesigned in event-driven manner, so that only minimum compulsory

function blocks will be executed. This is able to improve application efficiency significantly.

(6) Finally, a table of terminology mapping table is provided as migration suggestions:

IEC 61131 Terminology	IEC 61499 Terminology
Configuration	System Configuration
Resource	Device
Task	Resource
Program	Application

Fig.8. Terminology Comparison Table

B. *Limitations*

Although industrial adoption of IEC 61499 is just a matter of time, there are still some concerns needed to be addressed before IEC 61499 can be deployed widely.

First important issue is concurrency. In IEC 61131 implementations, SFC can have multiple states active simultaneously. But under current definition of IEC 61499, only one state is allowed to be current active state in ECC. This prevents direct conversion of the 'old' FBs to the 'new' ones.

Secondly, there is no global memory sharing over the function blocks. Implementing access to global data via a "storage" function block makes the design messy and unreadable.

Also the advantages of distribution and reconfigurable ability are not benefiting small single processor systems. In a tiny application, IEC 61499 function block controller is not cost-effective compared to current micro PLCs although the benefits are pretty obvious when the project size scalar multiple times.

Last but not the least, as per IEC 61499, the function block is only executed once and will not be executed again until another event input is triggered. In current PLC setup, when a logic rung is not executed in a scan, the output of this logic rung will be automatically turned off. This will not happen in ladder logic encapsulated to IEC 61499 function blocks. When a different input event triggered compared to previous scan, the function block will only update data associated with that event. The data not associated to that event input will not be updated and all other irrelevant output data for this scan may not be updated until some input events rise.

V. CONCLUSIONS

The feasibility of migration from PLC implementation to IEC 61499 function blocks is analyzed in this paper. A sample test bed is programmed in both IEC61131 PLC language and IEC 61499 function blocks. An equivalent system in IEC 61499 function blocks is introduced under the new event-driven design concept. So far, only simulation of IEC 61499 was accomplished and the transformation from IEC 61131 to IEC 61499 is not 100% equivalent. Some limitations of current IEC 61499 standard are discussed.

This research will be continued towards running the obtained IEC 61499 codes on the controllers connected to the real conveyor system. Also improvement of the current IEC 61499 standard needs to be investigated to improve efficiency and help migration IEC 61131 code to IEC 61499 function blocks.

VI. REFERENCES

[1] IEC 61131-3, "Programmable controllers - Part 3: Programming languages", International Standard, Second Edition, 2003

[2] IEC 61499, "Function Blocks", International Standard, First Edition, 2005

[3] C. Gerber, H.-M. Hanisch, and S. Ebbinghaus, „From IEC 61131 to IEC 61499 for Distributed Systems: A Case Study", EURASIP Journal on Embedded Systems, Volume 2008, Article ID 231630, 8 pages, doi:10.1155/2008/231630

[4] C. Sünder, M. Wenger, C. Hanni, I. Gosetti, H. Steininger, J. Fritsche, „Transformation of existing IEC 61131-3 automation projects into control logic according to IEC 61499", IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg, Germany, September, 2008

[5] G. Black, V. Vyatkin, "Intelligent Component – based Automation of Baggage Handling Systems with IEC 61499", *IEEE Transactions on Automation Science and Engineering*, vol. 6, 2008, in press

[6] Vyatkin V., Salcic Z., Roop P., Fitzgerald J., "Information Infrastructure of Intelligent Machines based on IEC61499 Architecture", *IEEE Industrial Electronics Magazine*, 2007, 1(4) pp. 17-29

[7] Vyatkin, V., "IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design", International Society for Automation, 2007, Triangle Park, NC, USA

[8] FBDK website: http://www.holobloc.com/

[9] ISaGRAF Website: http://www.arcom.com/products/pcp/pcp16.htm

[10] FBench website: http://www.ece.auckland.ac.nz/~vyatkin/fbench/

[11] OOONEIDA website: http://www.oooneida.org

[12] V. Vyatkin, J. Chouinard, "On Comparisons the ISaGRAF implementation of IEC 61499 with FBDK and other implementations", 6th IEEE International Conference on Industrial Informatics (INDIN'08), Daejeon, Korea, July 2008, Page(s):289 - 294