9th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS, Suceava, Romania, May 22-24, 2008

Modeling Manufacturing Systems Using the IEC 61499 Standard

Valentin VLAD [1], Cristina Elena TURCU[2]
"Stefan cel Mare" University of Suceava
str.Universitatii nr.13, RO-720229 Suceava
[1] *vladv@usv.ro*, [2] *cristina@usv.ro*

***Abstract*—The continuous evolution in hardware and fieldbus communication led to a growth of complexity in these areas and, as a result, the need for new standards, new modeling and development tools for distributed systems has been appeared. IEC 61499 provide for engineers a generic distributed modeling platform which can one hand simplify the modeling of distributed systems and on the other hand diminish the differences in modeling between business and industrial systems. This paper describe a modeling application based on the IEC 61499 standard and the modified MVC pattern, for a manufacturing system in miniature, consisting of two machining lines, one assembly line and the transport system.**

***Index Terms*—Distributed Control, Function Block Methodology, IEC 61499, Manufacturing Systems, MVC Pattern.**

I. INTRODCTION

In the last decades significant changes in the manufacturing environment have been noticed: moving from a local economy towards a global economy, with markets demanding for products with high quality at lower costs, highly customized and with short life cycle, leading to mass customization. To remain competitive, the companies search to answer more close to the customer demands, by improving their flexibility and agility while maintaining their productivity and quality.

In parallel, the continuous evolution of technology often requires the updating and integration of existing systems within new supervisory environments, to avoid their technological obsolescence.

Classical manufacturing systems are characterized by a good production optimization but a weak response to change, mainly because of the rigidity and centralization of the control structure. In these circumstances, the challenge is to develop manufacturing control systems with autonomy and intelligence capabilities, agile and fast adaptation to the environments changes, more robust against the occurrence of disturbances, and easier integration of manufacturing resources and legacy systems [Leitão, 2004].

Many approaches have been proposed in the academic literature to deal with the reconfiguration problem and internal and external disturbances in manufacturing systems. The well-known approaches are biological manufacturing systems (BMS) [Ueda et al., 1997], holonic manufacturing systems (HMS) [HMS, 2008], fractal manufacturing systems [Warnecke, 1993] and virtual cellular manufacturing (VCM) [Drolet s.a., 1989].

II. THE IEC 61499 STANDARD

IEC 61499 is a standard developed by IEC (International Electrotechnical Commission) which define a software paradigm built on the basis of FB (Function Blocks) that encapsulates data along with its behavior in a form that is similar to physical entities, i.e. electronic circuits or hardware elements. The initiation of the development process of this standard dates back to 1990 and reside in IEC Technical Committee 65 proposal that function blocks would be a generic concept that could be applied to a wide range of standards i.e., PLCs, Smart Devices, Fieldbus protocols, building management systems and thus could be considered as the essential element of decentralized and distributed IPMCSs (Industrial-Process Measurement and Control Systems) [Christensen J., 2000]

The IEC 61499 Standard defines an open architecture for the next generation of distributed control and automation. The key elements of distributed control architecture under IEC 61499 are *application*, *device* and *resource*. An application is a related set of functions that must talk to each other to fulfill a control task. A device is a control unit having one or more processors. It interfaces to the physical I/O and also communicates with other devices on the network. A resource is essentially a processor on which part of a distributed application will run.

The programming unit of the IEC 61499 is the *function block* (FB). It is the basic building block from which entire applications may be built. There are three types of function blocks: basic function blocks, composite function blocks and service interface function blocks.

A *basic function block* (Figure 1a) executes an elemental control function, such as reading a sensor or setting the state of an actuator, and contains *algorithms and an execution control chart* (ECC). Basic function blocks may be combined

together in a *composite function block*, to encapsulate a higher-level control function. The *service interface function block* provides the communication services among devices.

A composite function block (Figure 1b) contains other composite function blocks and/or basic function blocks.

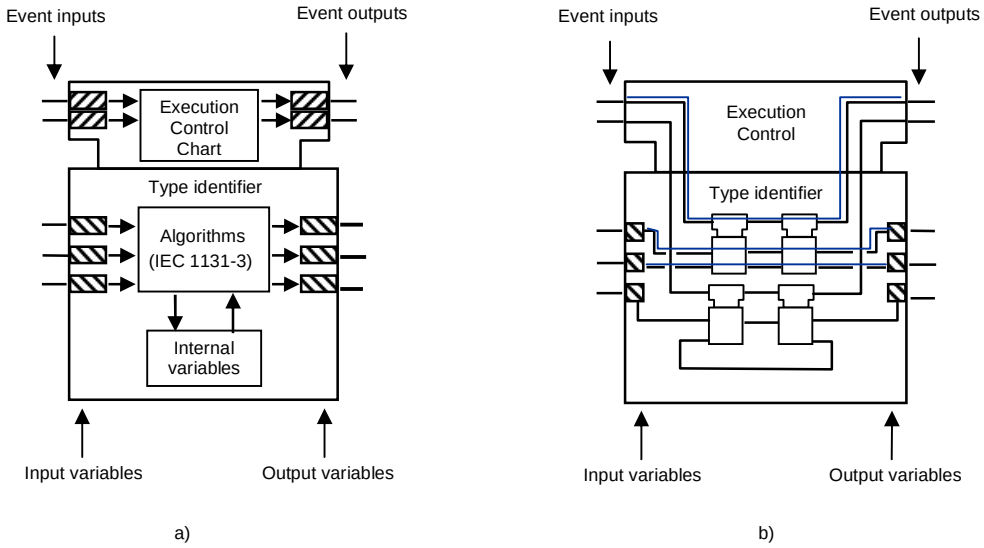


Figure 1. a) basic function block; b) composite function block

The standard is based on an *event-based execution model*. Each function block has event inputs and outputs as well as data inputs and outputs. In a basic function block, the execution of an algorithm is triggered by the occurrence of an input event. The executed algorithm produces from the input data new output data. When the algorithm has finished, an output event is generated. This output event might then be the input event to another function block [Vyatkin, 2006].

III. MVC PATTERN

MVC (Model – View – Controller) is an architectural pattern used in software engineering which defines a way of breaking an application, or even just a piece of an application's interface, into three parts: the model, the view, and the controller. In this manner, a component may be modified without affecting the others.

The traditional MVC pattern (Figure 2a) was developed to address separation of concerns in user interfaces for object-oriented systems. In this pattern, the *model* contained data whose values were independent of their graphical representation; the role of *view* was to provide a specific textual or graphical rendering of some or all of the data in the model, and the *controller* managed the user interactions with the model and view.

In [Christensen, 2000] is presented a possible adaptation of this pattern for use in the modeling, simulation and testing of industrial-process measurement and control systems (IPMCSs) in the IEC 61499 context, by modifying the definitions for model, view and controller. Thus, the *model* is defined as a function block that represents the time-dependent logical behavior of the system or device being controlled. The *view* is defined as a FB that represents the graphical display associated with one or more model types, and the *controller* is replaced by a function block that encapsulates the control functions to be performed on one or more instances of associated *model* types. The controller present appropriate event and interfaces for integration of its functions with those of other *controller* blocks.

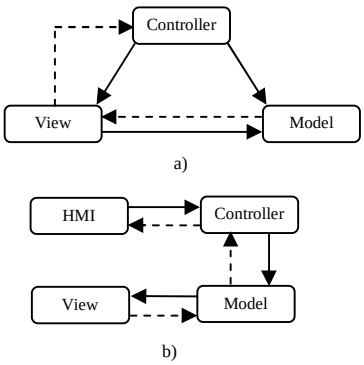


Figure 2. MVC Pattern a) Classic form; b) Modified framework

Whether in the classical MVC framework a controller represents the functions that may be performed by a human interface element to modify the data in the *model* or the appearance of the model presented in the *view*, in the new framework user interaction is represented by a HMI (Human/Machine Interface) element, which is also a function block (Figure 2b).

IV. SOFTWARE APPLICATION

In this section a simple IEC 61499 compliant application for modeling a manufacturing system in miniature is described. The manufacturing system is consisting of two machining lines, one assembly line, manipulators and conveyors.

The application is developed using the development kit for functions block FBDK licensed by Rockel Automation Company [FBDK, 2008].

FBDK is an effort of the Holonic Manufacturing Systems consortium and allow the users to define function blocks and to use built-in and/or user defined FBs to design elements of a distributed control application.

Using FBDK each FB Type (FBT) can be translated to a class in Java. Then each of the FBs, whether user-made or built-in, can be tested for the reliability of its designed functionalities. Moreover, applications, both stand-alone and distributed can be run along with a graphical interface for visualization and parameter variation.

A. General presentation

The application simulates the manufacturing process of products composed by two components that are first machined and then assembled in an end product, as may be observed from flow sheet illustrated in Figure 3. Raw materials are provided from *Store 1* towards *Machine 1* and *Machine 2* to be machined in Type A respectively Type B products. The resulted components are then transported to an assembly system where they are assembled into end products. The finished products are finally transported and stored in *Store 2*.

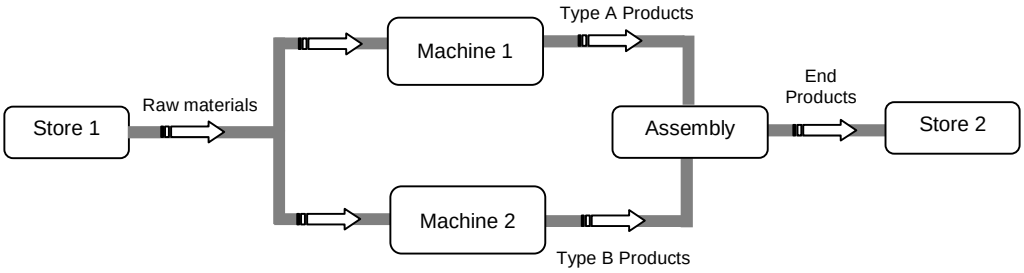


Figure 3. The manufacturing flow

The architecture of the application is based on the MVC pattern, adapted for IEC 61499 context, previously described. It is divided into four main components:

(a) the *graphical interface* (the view) through that the user may graphically observe the evolution of the system,

(b) the *HMI interface* through that the user may interact with the system and may look for the values of system's sensors,

(c) the *control component*, which manage the operation of the system in correlation with user's actions and

(d) the *model*, represented by function blocks that encapsulate data and behavior for each element of the system (manipulators, conveyors, machines and assembly devices).

B. Description of implementation

The developed system is composed by three devices: *Graphic*, *Control* and *Commands*, each of them being composed from resources, and resources on their turn from interconnected function blocks encapsulating algorithms. Each device is related with a window of the application, with the same name, as can be seen in Figure 4.

The *Graphic* device contains resources and function blocks that represent the behavior of the elements of the system (machines, transport elements) and allow graphic visualization of the system evolution. Also, it contains FBs for displaying a text or a background color.

The *Control* device provides a HMI interface for user interactions and encapsulates resources and function blocks which assure the control of elements involved in the simulated manufacturing process. By means of the HMI interface an user may trigger the execution of a waiting command (using the START button), may switch between *Auto* and *Manual* mode and may monitor the values of the sensor associated with machining, assembly and transport elements. Also, the window associated with the *Control* device provides the values of two counters: one for number of waiting commands and one for finished products.

The *Commands* device, related with the *Commands* window of the application, simulate an application running on a computer, through that someone may create and send commands for products toward the manufacturing system, specifying the type and the amount of products to be realized.

When the system receive a command from the *Commands* device, if the system is in the *Manual* state, the *Waiting command* led start to blink and the counter for *Waiting commands* is updated. The system will remain in this waiting state until the START button is pressed. Then the execution of waiting command is triggered and, from *Store 1*, two pieces of raw material for each end product are delivered by the manipulator *Robot 1* toward *Machine 1* and *Machine 2*.

In the simulated process the type of products is represented by the color of the component machined by Machine 1 (a ring). This color is extracted from the command sent by the user using the *Commands* window.

The process of delivering raw materials from the Store 1 will continue until the number of waiting commands became zero. After the assembly process, the end products are transported toward Store 2. Each time a product is stored in Store 2, the Finished products counter from Control window is incremented.

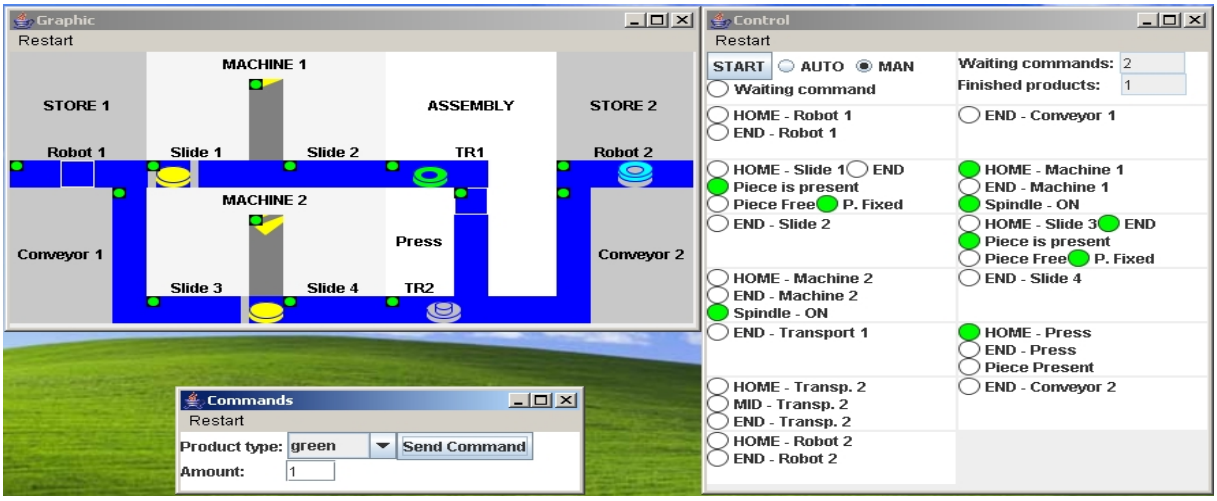


Figure 4. Application windows

Significant function blocks

In Figure 5 are presented the FB interconnections inside the START_CTRL resource. The purpose of this resource is to take up the commands sent by Commands device, launch them in execution if the system is in the Auto state or activate the blinking of Waiting command led if the system is in the Manual state. When the START button is pressed the resource start the execution of the waiting commands.

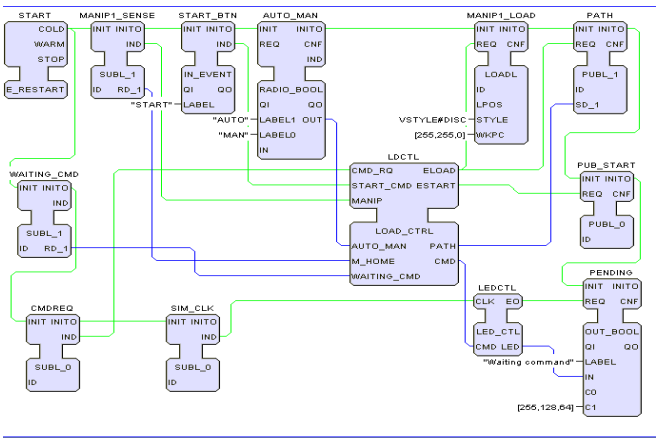


Figure 5. FB interconnections inside the START_CTRL resource

The controller of the START_CTRL is an instance of the LOAD_CTRL basic function block (Figure 6). In Figure 6a is depicted the FB's interface with events and data I/O, and in Figure 6b is presented the Execution Control Chart (ECC) of the FB.

The data inputs of the FB are the position of the Auto/Manual switch, the HOME sensor of the manipulator *Robot 1* from *Store 1* and the counter of products waiting to be fabricated. The FB has two boolean data outputs: PATH – indicating the path that the piece of raw material, currently transported by *Robot 1*, must follow (to Machine 1 or to Machine 2), and CMD – which enable or disable the blinking of the *Waiting command* led.

The controller receives three kinds of events: (a) events from the *Commands* device, when someone launch a new request for products, (b) events from the FB associated with the START button (requests to trigger the execution of a waiting command) and (c) events from the FB associated with the manipulator *Robot 1*, indicating the moment when the manipulator is in the HOME position.

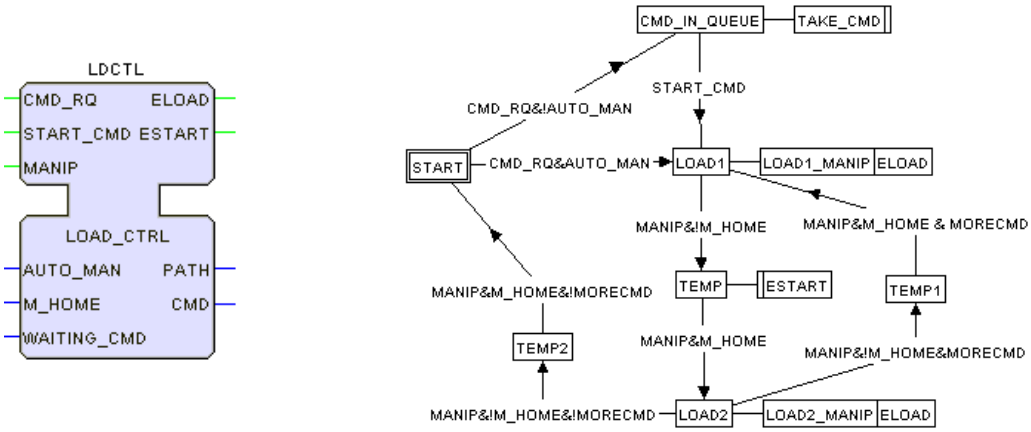
When a CMD_RQ event is received (from the *Commands* device) the controller analyses the position of the *Auto/Man* switch. If its current position is *Auto* the command is launched in execution by an output event (ELOAD) generation. Otherwise, if the switch is in the *Manual* state, the presence of a waiting command is retained in a boolean internal variable and the output variable CMD get the ON value for activating the blinking of the *Waiting command* led.

Pushing the START button generates a START_CMD event for the LOAD_CTRL function block. This event has significance only there is a waiting command and leads to stop the blinking action of the indicator led and the type and the amount of products to be start the command execution by loading *Robot 1* with a raw material piece. When *Robot 1* comes back in the HOME position after delivering the first raw material piece (which was routed to Machine 2), it is automatically loaded with a new raw material piece to be routed toward Machine 1. If there are no other commands to be served when the manipulator comes back again in the HOME position, the controller goes in the START state, waiting a new command (Figure 6b).

Another important function block is the QUEUE_CLR FB that implements a queue for the commands received from the

Commands device. The commands are stored in the queue until its are launched in execution. In this queue are retained fabricated, for each command. While the system is serving a command, each time the execution of a new product is started, the counter for the amount requested by the command is decremented. When the value of the counter became zero a new command is extracted from the queue. The system stops when there is no any command in the queue.

The function blocks considering in the design phase are abstract representations and may be developed in different device types (PLC, microcontroller based stations, etc.). But functions implemented for a function block are not necessarily associated one-to-one to the device. Also, other applications, such as a supervisory system may be connected to the system.



a) b)

Figure 6. LOAD_CTRL function block
a) Interface; b) Execution Control Chart - ECC

V. CONCLUSIONS AND FUTURE WORK

This paper presents a basic model of a simple application for modeling manufacturing systems, using the IEC 61499 standard. The application represents a first step in a more complex project dealing with the reconfiguration problem of the manufacturing systems.

The IEC 61499 standard, recently adopted (2005), is currently analyzed and employed in distributed applications by many researchers, towards to be extended such as to satisfy at the best hand the requirements for development of distributed systems. In Romania, the researches in this area are in an incipient phase.

The encapsulation offered by IEC 61499 supply a modular development which makes easy the redesign or design of future systems. Some benefits of this IEC 61499 approach are: (a) intuitive programming, (b) reutilizing of function block and (c) easy debugging.

Future work is intended to add new functionalities to the application such as connectivity to a database, possibility of system remote monitoring through a web browser and supplementary control for system auto reconfiguration, when a device brakes down. Also will be examined the possibility to deploy the application on physical devices for controlling real machining and assembly systems existing at the University of Suceava.

REFERENCES

[1] J.H.Christensen, Basic Concepts of IEC 61499, available at http://www.holobloc.com

[2] J.H.Christensen, Design paterns for sistem engineering in IEC 61499, Otto-von-Guericke-Universitat Magdeburg Germany, 22-23 March 2000, 63-71.

[3] Drolet, J.R., Moodie, C. L., Montreuil B. Scheduling factories of the future, J. Mech. Work. Technol. 20 (1989) 183–194.

[4] FBDK. Function Block Development Kit – available at http://www.holobloc.com

[5] HMS. Holonic Manufacturing Systems Consortium - web site. http://hms.ifw.uni-hannover.de, 2008.

[6] T.Hussain, G.Frey, Developing IEC 61499 Compliant Distributed Systems with Network Enabled Controllers, Proceeding of the 2004 IEEE Conference on Robotics, Automation and Mechatronics, RAM-2004, Singapore, pp. 613-618, Dec. 2004

[7] Leitão, P. An Agile, Adaptive Holonic Architecture for Manufacturing Control. Phd Thesis, University of Porto, 2004

[8] Ueda, K., Vaario, J., Ohkura, K. Modelling of biological manufacturing systems for dynamic reconfiguration, Ann. CIRP 46 (1997) 343–346.

[9] Vyatkin, V. IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design, Instrumentation Society of America, USA, July, 2006.

[10] H.J. Warnecke, The Fractal Company, Springer, Berlin, 1993.

[11] Calin CIUFUDEAN, George MAHALU, Availability of Flexible Manufacturing Cells, Advances in Electrical and Computer Engineering, Suceava, Romania, ISSN 1582-7445, No 2/2001, volume 1 (8), pp. 15-18

[12] Cornel TURCU, Cristina TURCU, Facilities in DesignerAX–an All-Purpose Automation ActiveX Framework, Advances in Electrical and Computer Engineering, Suceava, Romania, ISSN 1582-7445, No 1/2003, volume 3 (10), pp. 17-20

[13] Leonard IURESCU, Using a Multiagent System for Controlling a Manufacturing Cell with a Robot, Advances in Electrical and Computer Engineering, Suceava, Romania, ISSN 1582-7445, No 1/2005, volume 5 (12), pp. 46-50

[14] Cornel TURCU, Cristina TURCU, Alin Dan POTORAC, Adrian GRAUR, Considerations over the Data Acquisition System for Energetic Power Consumption, Advances in Electrical and Computer Engineering, Suceava, Romania, ISSN 1582-7445, No 2/2002, volume 2 (9), pp. 5-9