RECONFIGURATION OF MECHATRONIC SYSTEMS ENABLED BY IEC 61499 FUNCTION BLOCKS

A.R. Sardesai, O. Mazharullah and V. Vyatkin
Department of Electrical and Computer Engineering
University of Auckland, Auckland, New Zealand
v.vyatkin@auckland.ac.nz

Abstract

This paper discusses a new approach to the development of industrial manufacturing control systems. In industry, there is an increasing need for more intelligent and agile control systems which are able to be flexible in their operations. Through use of the new IEC 61499 standard, a new design methodology for modeling industrial control systems has been developed with these goals in mind. A design pattern for the control of automation devices has been created and tested. In addition, a distributed intelligent automation system has been developed in which different devices are able to interact, communicate and adjust to their respective environments.
In order to view and test the developed system, a web based Human-Machine Interface (HMI) has been created. This HMI is able to be run on any internet enabled device, including Personal Digital Assistants (PDAs) and mobile phones. Through this HMI the workflow of the system can be viewed and modified.

1. Introduction

In today's industrial manufacturing environment there an increasing need for flexible, agile and adaptable manufacturing systems. Traditional manufacturing systems have typically relied upon a centralized hierarchy of programmable logic devices. This hierarchy is usually fixed or “hard-wired” [Vyatkin, 2006]. As a result, in order to reconfigure the system, the plant must be shut-down and the components must be rewired and reprogrammed. From a purely economic standpoint this is undesirable.
A flexible and agile manufacturing system must be able to adapt and reconfigure based on changing environments. They must be reusable across different manufacturing systems and must be able to be deployed across a wide array of micro controllers. It must be able to react in real time while still being safe and reliable.
For this reason, there needs to be a paradigm shift from centralized PLC control to more distributed control architecture. Each component within the system must have its own distributed controller which is able to communicate with other devices while still being able maintain its performance characteristics. The lack of any central supervisory control also means that there is no single point of failure. Components can be removed while the rest of the system adapts and continues its operation.

2. IEC 61499 standard

In order to achieve the goals of flexible and reconfigurable manufacturing systems, new system development architecture has been standardized. The IEC 61499 standard is an open component based architecture designed for the development of distributed control and measurement systems [IEC61499, 2005]. It describes how a distributed control system can be created through the interconnection of event driven software components. An IEC 61499 application can be embedded within a micro controller such that it is able to provide real time control of a device.
The fundamental unit of software encapsulation is the Function Block. Function Blocks are reusable and are able to encapsulate intellectual property. An important point to note is that Function Blocks are event driven. Main value of being event-driven is that the blocks do not depend on any external schedule. So, a block completely encapsulates the desired function and will show same behavior anywhere. When an event occurs in the environment, a user can choose to have a Function Block react to this event and carry out a particular calculation or provide some output. In this way, networks of Function Blocks can be developed to fulfill particular applications. It is important to note that the IEC61499 standard provides a directly executable specification. Function block systems are made to be run with real system, not to merely model them.

3. Function Block Development Kit (FBDK)

The most widely used tool used in the development of function block systems is Function Block Development Kit [FBDK, 2006]. It provides a graphical interface

through which IEC 61499 compliant systems can be created and tested. Function Block Runtime (FBRT) is a runtime platform which allows systems created with FBDK to be run on any Java enabled platform.
Any micro controller which has a correctly installed FBRT is able to run function block systems. As a result, the FBRT system is a step towards middleware in industrial automation. Linked to this is the notion of portability. There is no need for proprietary languages and development systems which can only be run on one type of controller. This also facilitates the intercommunication between different micro controllers such that larger and more complex intelligent systems can be developed.

4. The University of Auckland Test Bench

The University of Auckland test bench consists of two FESTO mechatronic stations [Fig 1]**.** The purpose of the setup is to simulate industrial manufacturing systems and to test various system configurations.

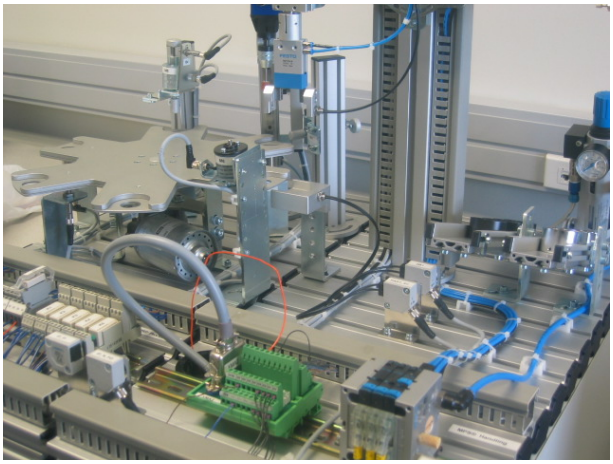

Figure 1 – The University of Auckland Test Bench

Station 1 consists of a rotating table, three presence sensors, a hole checker and a drill. In its standard operation, the table will rotate whenever a piece is placed on it. The drill checker will see if there is a hole in the work piece, in which case the drill will simply ignore the work piece. If there is no hole in the work piece, the drill will be activated to create a hole. The work piece is then pushed onto Station 2.
Station 2 consists of a pneumatic arm, a presence sensor, a reflection sensor and two trays. When a work piece is present, the arm picks it up and moves it into one of two trays depending on the color of the piece. A faulty work piece is defined as one which has no hole or is black.
In order to control the test bench, one Elsist Netmaster micro controller and a TCS MO'Intelligence micro controller were installed as shown in Figure 2. Both of these controllers are able to run Function Block systems. In addition, the stations provide all inputs and outputs as binary values that can be easily manipulated using the two aforementioned micro controllers. The function block controllers were also connected to the FEC PLC controlling the pneumatic arm.

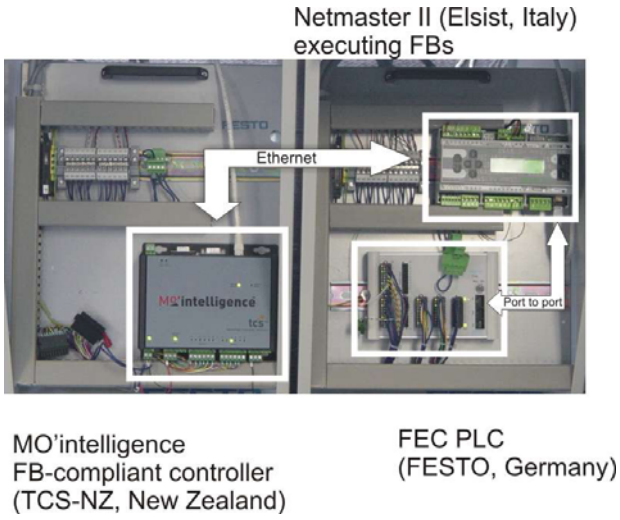

Figure 2: Distributed hardware devices controlling the test bench.

5. Implementation of Distributed controllers using IEC 61499

The problem of efficient design of distributed controllers is an open research subject addressed in a number of works, in particular in [2] and [4]. In this work, in order to facilitate the development of intelligent mechatronic actors, a standard software structure called the 'Interface' Function Block has been developed. [Fig 2].

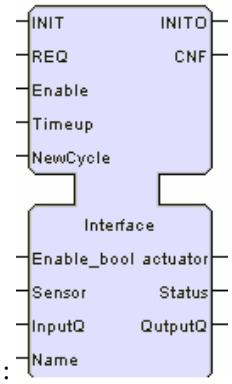

Figure 3: The Interface Function Block

The intent of developing this software module was to create a kind of ‘standard template’ structure for a function block that controls a basic function of any arbitrary device, for example, rotation of the table, or linear movement of the drill. As long as the precondition of the function can be expressed as a predicate on Boolean inputs, the 'Interface' function block can be used. In the case of a device implementing more than one function, several 'Interface' function blocks can be cascaded to provide the appropriate functionality.
When the relevant sensor event occurs in the environment, the device controlled by the ‘Interface’ function block will be activated for a particular length of

time. This is achieved by setting the value of 'actuator' to high when the 'REQ' event is received and the value of 'Sensor' is high. It will stay high until the 'time up' event is received. The 'Interface' function block cannot be reactivated until the 'NewCycle' event is received. This prevents premature reactivation of the device which may hinder the safety of the overall system. The Enable input to the interface function block allows individual components to be enabled and disabled without affecting the entire system.

As a result, a software module that can be used to control the operation of any mechatronic device has been created. Through simple manipulation of events, a device can be operated. When many devices are cascaded into a larger system, more complex and interconnected operations can be carried out.

Through the use of the 'Interface' function block, distributed system architecture can be achieved. Each device has its own controller. This avoids any single point of failure and means that each device can be independently tested and serviced. A system with multiple 'Interface' function blocks has the advantage of performing multiple workflows through the manipulation of the 'Enable' inputs. Hence, flexibility in manufacturing systems has been achieved.

6. Intelligence in distributed system

The mechatronic test bench system model is designed using distributed system architecture. The distributed structure allows the system to be flexible and re-configurable. Replacing or repairing a faulty component in an assembly line forces it to be halted, incurring a large cost for the factory. This was an important issue to be considered when developing intelligence in the system.

As discussed previously, the 'Interface' function block is the base element used to create a distributed system. However, this alone is insufficient to provide the intelligence. For this reason, 'Intelligent Agents' have been introduced. Each device, controlled by an 'Interface' function block, may have an associated intelligent agent which reads the current system state and its associated data to manipulate actuators in a particular fashion. Each intelligent agent can set the 'InputQ' of an 'Interface', thus manipulating its actions.

There are two possible ways to create intelligent agents; each of which is suitable for system engineers of different back-grounds. These are both discussed below through means of an example.

1.1. Combinational Logic Approach

As discussed earlier, the test bench has a drill measurer and a drill. In a normal scenario, the hole checker checks to see if a work piece has hole in it. If this there is no hole, the drill makes a hole in the work piece. However, if the drill measurer is disabled or has malfunctioned it will not provide this information, so the system must reconfigure and adapt. The workflow must change such that every work piece is drilled.

The logic which explains this can be represented in a form of a truth table [Table1].

Drill Measurer Enabled	Drill measurement failed	InputQ (drill)
True	True	True
True	False	False
False	X	True

Table 1: Truth Table for Combinational Logic

This truth table can be converted into combinational logic, a common approach used in circuit design by electrical engineers. After this, a combination of function block logic elements can be used to create an intelligent agent for the drill. [Fig 3]

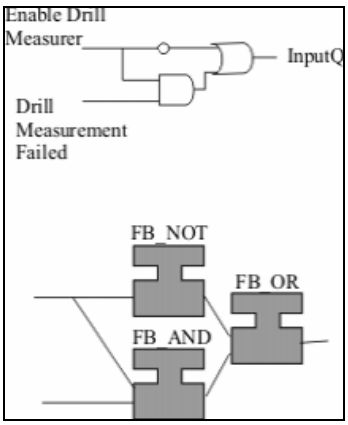

Figure 4: Combinational Logic in Function Blocks

1.2. Programming Approach

The combinational logic approach can be a tedious task and the design can get confusing with increasing complexity of dependencies. Hence, a simpler but more programming based approach is used in the system to design the intelligent agent of the sorting station. In this approach, the functionality of an intelligent agent is captured in a basic function block in form of an algorithm.

In the sample scenario, the sorting station should sort between faulty and non faulty, depending on whether or not the work piece was drilled properly.

When the drill is working, the station should sort the pieces as non-faulty. But if the drill itself is faulty, Station 1 must use the hole checker to determine whether or not a work piece already has a hole in it. Again, only the work pieces with existing holes will be classified as non-faulty. If both the drill and the hole checker are non-functional, then the system must divert all pieces into the faulty tray.

This functionality is to be captured in a form of truth table [Fig 2]. If a work piece is to be diverted to the faulty line, then the output of the intelligent agent must be 'false'.

Drill Enabled	Drill measurer Enabled	Drill check passed	InputQ (Sort)
True	X	X	True
False	True	True	True
False	True	False	False
False	False	X	False

Table 2: Truth Table for the Sorting of Work Pieces

This functionality is captured in a basic function block with an algorithm (see Figure 4).

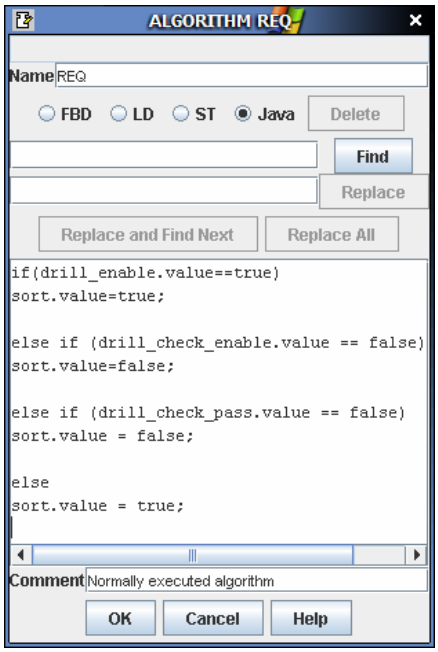

*Figure 5:*JAVA *algorithm implementing the sorting logic encapsulated into function block.*

7. Human Machine Interface (HMI)

HMI provides a user or operator with the current state of a system or process. It also facilitates the control of a process.

After the development of a distributed control system for the mechatronic test bench, it was necessary to create a HMI which could control this test bench. The HMI was initially created using standard FBDK visualization components. The functionality of this HMI system and its integration with the system control was tested thoroughly. However, the functionality provided by this single PC based HMI is insufficient for use in an industrial production environment. The FBDK based visualization can only be run from a computer on which either FBDK or FBRT is running. Also it is necessary to decide which computers will be displaying these HMI elements before launching the system configuration on the embedded platforms. It was not possible to move the HMI or visualization to any other computer after launch of the initial system configurations. This is a major drawback of this system.

Portability of the HMI was considered as an important aspect in this project. Two solutions were considered. The first solution was to use the standard visualization elements in FBDK and create a system configuration which could be launched at any time on any PC and control the system through network. The second solution was to create a new webpage based HMI system from scratch.

The first solution requires that the user have a computer with FBDK or FBRT installed. In addition, every computer which needs to view this HMI will have to upload and calibrate the new system on its FBDK. This also causes a new restriction that every computer will have to upload the HMI system again if there is any update in it. The second solution of webpage based HMI eliminates all these issues. Hence, this solution was chosen for development.

1.3. Development of a Web Based HMI

The web based HMI needs to provide a dynamically re-configurable and an interactive user interface. Java based applet programming was chosen. An applet is a Java program which is run on a web page by including it inside the source code of the page.

This applet programming method was chosen for two main reasons. Firstly, FBDK itself is written in Java. Hence, creating a Java based HMI system was a favorable solution. Secondly, the applet solution provides most of the networking and GUI drawing features of the base Java language. This helps an applet to be dynamically re-configurable through the internet with no changes to the HTML web page.

The development of this applet based system consists of two important parts: the function block representation of the HMI interface and the communication protocol between an IEC 61499 system and a web browser.

1.4. Function block development for the HMI

The HMI interface is displayed through an applet on a web browser. It was necessary to create function block representation of this interface to make it compatible with the IEC 61499 standard.

Four elements were identified to be essential for an implementation of a HMI system. These four elements are ‘Label’ (for textual information), ‘Button’ (for an event input from user), ‘Check Box’ (for Boolean input from the user) and ‘Radio Button’ (for Boolean output to the user). For each of these four elements, a function block was created.

The function blocks have position variables as inputs such that they can be accurately represented on a web page. They also have event and variable inputs and outputs which represent an action on the corresponding element of the web page. A network of these function blocks can then be integrated with any standard IEC 61499 system for the display of a web based HMI.

1.5. Communication

The communication between the applet and FBRT is handled by a function block resource, which acts as an proxy between these two. The function blocks update their information to this resource. This information is then passed on to the applets through a pre-defined protocol over the internet. Also, the resource reads information from the applet elements and passes them on to the respective function blocks. Please refer to Figure 5 for an illustration of this information flow. An important part of this HMI development was to generate a network efficient, yet fully functional protocol for the communication between FBRT and the web applet.

8. Development of a Tool for Generation of a HMI

After developing the applet and the compatible Function blocks, the main challenge was to give the exact position parameters to the function blocks for the expected display. It was very time consuming for big display interfaces. Hence it was decided to develop a tool for generation of this interface.

This software tool is written in Java. It allows the developer to create, save and modify HMI interfaces. It gives the developer control over type, position, size and the name of an element. After creating a suitable interface in the tool, it allows the information to be exported in XML – based format of Function Blocks. The tool can write an XML file readable by FBDK. The XML file consists of a set of function blocks with pre-defined variable inputs for positions. This provides an efficient way of creating a HMI by giving a visual preview of the interface at development time. This considerably reduces the development time as the need of writing the visual properties of the function blocks is eliminated.

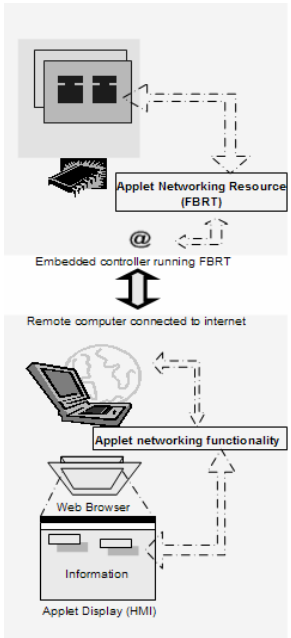

Figure 6: Communication Flow between FBRT and Applet

9. HMI for the Mechatronic test bench

After creating the web based HMI system and the tool to generate the function block network, a web based HMI was created to control the Mechatronic test bench. The HMI consists of two pages. The first page gives information to the user about the current workflow in the system and the state of individual components. The second page is an administrative page which gives control over the functionality of the test bench.

There are two modes possible. In automatic mode, the auto re-configurability of the system is enabled. In this mode, two of the components can be enabled and disabled to check the function of the intelligent agents in the auto re-configuration of the system. This workflow is also textually displayed on the information page. In manual mode, individual components can be activated with the buttons. This mode can be used in testing individual components for any faults. The final system is tested to run successfully on 4 simultaneous web connections in different modes for a prolonged period of time.

10. Future Work

One of the primary drawbacks of the IEC61499-based system engineering at present is the lack of formal verification tools. Function block systems tend to get complicated and there is a need for tools that can track and trace variables and events as a system executes. This is currently unavailable.

Linked to this is the notion of determinism. Industry demands systems that are reliable are able to produce particular values as output when a particular input is given. As more intelligence is developed, the deterministic nature of a system tends to be reduced. This needs to be guarded against.

Industrial automation systems also demand that there be guaranteed response times for all operations. Currently, the Function Block Runtime is implemented in Java which is not as fast and real time as conventional PLCs. Work must be done to improve the runtime execution rate such that as systems become more complex, their reliability is not decreased.

Conclusions

A model of flexible reconfigurable manufacturing system has been created under distributed control compliant with new international standard IEC 61499.

A design pattern for developing distributed systems was created. A control system modeled using this design pattern was proven to run successfully on a mechatronic test bench.

The fault tolerance of the system was improved by adding intelligent agents for auto re-configurability in addition to the standard distributed structure of the ‘Interface’ function block.

A web based, IEC 61499 compliant, HMI was developed and tested on the University of Auckland mechatronic test bench. A software tool was developed for a visual creation of this function block network.

References

[IEC61499, 2005] IEC61499, Part 1 “Architecture”: Function Blocks for Industrial Process Measurement and Control Systems, Standard, International Electrotechnical Commission, Geneva, 2005

[Vyatkin, 2006] Vyatkin, V., V.: *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design*, p.1-271, Instrumentation Society of America, USA, July, 2006, in print

[FBDK, 2006] Function Block Development Kit (FBDK/FBRT), http://www.holobloc.com/doc/fbdk/index.htm

[Vyatkin et al., 2006] Vyatkin, V., Hirsch M., Hanisch H.M., Systematic Design Of Distributed Controllers – and Their Implementation, 11th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2006), Proceedings, Prague, 2006