Migration of a PLC Controller to an IEC 61499 Compliant Distributed Control System: Hands-on Experiences

Tanvir Hussain and Georg Frey
Juniorprofessorship Agentbased Automation
University of Kaiserslautern
Erwin-Schrödinger-Street 12, PO Box 3049, D-67653 Kaiserslautern
{hussain, frey}@eit.uni-kl.de

Abstract - IEC 61499 ushers in a new trend of software development in the area of Industrial Process Measurement and Control System (IPMCS). This new standard simplifies the development of distributed IPMCS applications through inclusion of re-usability, encapsulation and modularity. IEC 61499, due to its close resemblance with Object-Oriented (OO) paradigm also paves the way to integrate modeling techniques like UML into the development process of the distributed IPMCS applications. Yet a remarkable challenge is there to integrate or re-design the systems, elements or components designed using a monolithic language like the ones on Programmable Logic Controllers (PLCs). In this work it is attempted to share the experiences faced while attempting to migrate a PLC controlled centralized laboratory application into an IEC 61499 compliant distributed control application. The targeted distributed control system consists of network-enabled controllers where the IEC 61499 compliant control sub-applications should run. It needs to be mentioned that the development process did not start from analysis of the specification of the instrument to be controlled but rather from a formal specification of it written in Signal Interpreted Petri Nets (SIPN).

Index Terms – IEC61499, Migration of PLC controller, Distributed control system.

I. INTRODUCTION

Introduction of IEC 61499 marks the beginning of a new era in the field of software engineering for Industrial Process Measurement and Control Systems (IPMCSs). Till now the majority of the IPMCS software have been monolithic, hardly reconfigurable or modifiable. This has been mainly due to the dominance of the Programmable Logic Controllers (PLCs) as controllers. But with the emergence of soft-programmable hardware and advanced communication techniques, i.e., fieldbuses and other standard protocols, complex distributed systems consisting of heterogeneous controller elements are becoming quite common in IPMCSs. Moreover, the current trend of market requires that the controllers should contain modifiable and reconfigurable software to allow the changes resulting from short product or technology, i.e., hardware, process flow or algorithm, life-cycles. The inadequacy of the existing software techniques for developing such complex distributed systems with the provision of flexibility and reconfigurability motivated the development of the new standard, IEC 61499.

The primary purpose of IEC 61499 is not as a programming methodology rather as an architecture and a model for distributed systems. This standard defines a number of models conducive for designing distributed systems. These models are defined in terms of Function Blocks (FBs) and allow describing distributed control systems (DCS) in an unambiguous and formal manner. Having a formal and standard approach for describing systems will allow the systems to be validated, compared and understood. It is then possible to translate this model into programming codes which can be embedded into physical devices of the distributed systems.

While the promising prospect of IEC 61499 in developing flexible and reconfigurable IPMCS applications attracts the developers to adopt it, a considerable amount of efforts are yet needed to find out convenient methodologies for the design and development of applications. Since, the FBs of IEC 61499 resemble the Objects of the Object Oriented (OO) software paradigm, the proposed design and development methodologies [1], [2] for IEC 61499 are also adopted from the OO realm. These design and development methodologies begin with the specifications of the system to be controlled or automated and go on till structuring the hierarchical arrangements and inter-relations between the FBs. During this process FBs are also grouped together to form device and resource models. In this hierarchical structure, applications and sub-applications can utilize the services of single or multiple entities according to the needs.

Since IEC 61499 based applications are supposed to replace the monolithic PLC language based ones, a reasonable migration methodology could prove less costly than the usual development approaches of beginning anew. For large and complex projects migration can save time and effort needed to reveal critical and fault-prone aspects of the applications as well as the interfaces and communication requirements.

In this paper such a migration methodology is elucidated through an example where a laboratory plant which had been controlled by a PLC was attempted to be controlled by an IEC 61499 compliant distributed control system deployed on network enabled controllers.

The following Section describes the laboratory plant. Section III describes the PLC controller and the formal description of the control algorithm in Signal Interpreted Petri Nets (SIPN) [3]. Section IV describes the migration methodology followed while converting the central control...
of the plant to an IEC 61499 compliant distributed one. The concluding section discusses the experiences gained through this work and forecasts the future developments.

II. DESCRIPTION OF THE PLANT

The plant for which the distributed control application has been developed was a didactic Modular Production System (MPS) from Festo [4] (cf. Fig. 1).

**Fig. 2 Schematic of the didactic plant**

The principal objective of the didactic prototype is to examine cylindrical work pieces for proper height and material type, drill a proper hole on each work piece and then sort them according to their material types. As the name implies, the plant consists of different modules. The modules are again grouped into four stations. Our concept of distributing the control is based upon the distinctive division of the plant into four stations. A brief description of the four stations is given below along with a schematic (cf. Fig. 2).

**Distribution Station:** This part of the plant consists of a pneumatic feeder module and a transfer module. The feeder module can push a single work piece at a time out of the magazine and sets the work piece on a place from where the transfer module can pick it up with a vacuum suction cap and after a 180° movement places it on the next station.

**Testing Station:** The testing station consists of a test spot, a lifting module, a thickness measuring module and a conveyor module. The test spot is equipped with 3 different types of proximity sensors, namely, inductive, capacitive and optical. The capacitive proximity sensor detects whether there is a work piece or not. The inductive proximity sensor detects whether the work piece is metallic or non-metallic and the optical proximity sensor detects, whether the work piece is black or not. The lifting module lifts the work piece from the test spot to the thickness measuring module, where it is tested for propriety of its thickness. A pneumatic cylinder mounted on the lifting module can push the work piece out of the lifting module. Depending on whether the thickness of the work piece is allowable or not, it is pushed out, respectively, to the conveyor or to the slider which slides it to the scrape area.

**Processing Station:** In this station runs the principal operation of drilling. The station consists of a rotary indexing table, a drilling module and a drilled hole checking module. The rotary indexing table has 4 places, 90° apart from one another, to hold work pieces. Placed on a particular place a work piece can appear on 4 positions as the table rotates. Position 1 is for receiving the work piece from the conveyor belt, and position 2 is allotted for the drilling operation. Position 3 is mounted with a drilled hole checking module and position 4 is for delivering the work piece to the next station.

**Storing Station:** This station is for storing the processed pieces according to their material types in different cylindrical magazines. The defective parts are guided through a slider to the scrape area. These functionalities are provided by the crane module and the transport module attached to the crane. With the help of its suction cap the transport module picks a work piece from the position 4 of the rotary table. Then the crane moves over the gantry to the particular magazine or to the slider depending on the work piece’s type and condition. The transport module then leaves the piece to the magazine or to the slider.

During the normal operation a work piece will be pushed out of the magazine by the feeder and transported to the test spot by the transport module. The lift module brings the work piece to the measuring module, and if the piece exhibits a tolerable thickness it is pushed by the cylinder mounted on the module to the conveyor, otherwise the lift moves down and pushes it to the slider which guides it to the scrape area. The conveyor guides a proper piece to the position 1 of the rotary table. The table then rotates till the piece comes under the drill module. After the drilling the rotary table moves further till it reaches the drilled hole checking module. After the piece is checked for correct drilling the table moves further so that the processed piece comes at position 4 from where the crane can pick it up with transfer module and then moves it to the proper repository. In case of improper drilling the crane puts the piece on the slider that slides it to the scrape area.

Apart from the normal mode, some other operating modes of the plant had been distinguished, namely,

- **Initializing mode:** Before starting the normal operation the modules need to be fixed on initial positions.
• **Clean-up mode:** After every initializing phase the operator can decide on whether the work pieces from previous operation that are left on the plant should be cleaned up or not. When a clean-up is desired this mode of operation comes into play.

• **Emergency stop mode:** Through pushing the emergency stop button the plant goes to this mode, where the modules along with the on-process work pieces are brought to safe halt.

• **Deadhead mode:** Before a normal halt the on-process pieces could be needed to be scraped without processing further. This functionality is offered in this mode when requested.

### III. SIPN-BASED PLC PROGRAM

SIPN are an extension of Condition Event Petri Nets. But its specialty lies on the fact that in it the influence of the environment on the system is based on signals instead of events. The transitions of SIPN are associated with a firing condition given as a Boolean function of the input signals while the places specify the output signals. Not only it enables developing comprehensive formal models of sequential and/or concurrent control algorithms, but also offers an intuitive and simple way to relating and/or transforming it to PLC programs. A tool called SIPN Editor is available for editing, visualizing, and implementing SIPN based PLC programs [5].

The didactic MPS plant was so far controlled by PLC and the control program was developed using SIPN. It is to be noted that the plant is divided into stations and further into modules and each of the modules provides a unique functionality. Therefore, for each of the modules a program subroutine was developed for each of the operating modes.

It is also mention worthy that, on each of the modes the operation of a single module is coupled with one or more of the other modules either in terms of work piece transfer or through information exchange. To realize these connections global variables were used. These variables represent the information regarding the on-process work piece and the other information that is needed to be exchanged. Moreover, for the overall process there need to be a sequential or hierarchical organization of the subroutines or program modules so that at any phase or mode of operation the sequential or concurrent execution of the subroutines yields the desired functionality.

It has been identified that the plant consists of 15 controllable entities. Furthermore it has been seen that there are – in most cases – several of these 15 entities performing identical actions. This observation simplified the task of modeling the control using SIPN as the development of a single SIPN control module suffices for all the identical entities.

To elucidate the development of the PLC program using SIPN, the SIPN for one of the modules is shown in Fig. 3. The SIPN corresponds to the feeder module and also to the other modules that perform 2-point movement, i.e., lift module, drill module, drilled hole checking module. This SIPN only shows the control algorithm for normal operative mode, later it was further extended to include the other modes of operation. The transitions T1, T2, T5, T6, T7, T8 and T9 are controlled by sensor inputs while the transitions T3 and T4 are fired on the basis of global variables. The SIPN ensures that the feeder will be pushed forward only when it currently stays in its backward position and vice versa.

The grey places ensemble the existence of hierarchical SIPN. When a token arrives at this place, an SIPN of lower hierarchy takes over the control and the token stays there until the finishing place of that inner SIPN is reached. In the SIPN of Fig. 3 two such places are there for ensuring that the operations, i.e., pushing the feeder forward and backward of the cylinder, are executed properly.

---

**Fig. 3 SIPN for normal operative mode of feeder module**

**Fig. 4 SIPN for overall plant operation**
The SIPNs for the modules that constitute a station are later coordinated using a SIPN of higher hierarchy. Then using a further SIPN the overall functionality is defined. Fig. 4 shows the SIPN corresponding to overall control functionalities. The notation of closely placed double-circles represents a call to a subroutine. As soon as the token enters the place denoted by the first circle the subroutine is called and as soon as the subroutine finishes its execution token is moved to the place denoted by the next circle. This notation makes the visualization and comprehension of the SIPN more transparent and simpler.

As far as the correspondence between the SIPN and PLC program is concerned, the modules’ functionalities are translated to subroutines and the functionalities of a station of the overall plant is realized through the main program that coordinates the subroutines.

IV. MIGRATION METHODOLOGY

The target of the migration depicted in this paper is to achieve an IEC 61499 compliant decentralized and distributed control system for the described didactic plant. The control should no longer be limited within a central controller or a central PC but should be scattered through a number of coordinated network enabled controllers. The network enabled controllers are those introduced in [6] and are called NETMASTER.

Development or modification of the FB Type (FBT) definitions as well as development and modification of the resource and device models can be carried out in any PC using Function Block Development Kit (FBDK) [7] and then can be deployed on the NETMASTER controllers using Function Block Run Time (FBRT) environment [7]. Due to being decentralized the operation of the controllers do not need a PC to be employed in the control process.

From the viewpoint of migration from PLC programs to an IEC 61499 compliant one this work resembles the work described in [8]. But a formidable difference lies in the approach that the aforementioned work considered the NETMASTER only for reading sensor data and activating the actuator inputs while the control was limited to a central PC. The control functionalities though were achieved through a distributed device model in accordance with IEC 61499 guidelines, yet those were not distributed to achieve decentralization.

The migration methodology adopted in the current work was carried out through a number of steps. The following parts of this section describe the steps and the concerning observations and results.

A. Distribution of the Hardware

In Section II we have seen that the plant can be divided into four stations and each of the stations contains more than one module. For the sake of distributed control we have used this distinctive division of the plant into stations. We have assigned a NETMASTER controller device for each of the stations.

The sensor inputs are connected to the digital input ports and the inputs to the actuators are connected to the digital output ports of the NETMASTER modules. The NETMASTER devices are connected to the Ethernet and thus can exchange information among them which is necessary for the coordinated operation of the stations.

Each of the NETMASTER modules has an LCD monitor and six buttons on it. These elements were used to achieve interaction. The current work was limited to implementation of the initializing and normal operating modes only.

The composition of the system model for the distributed control application is depicted in Fig. 5. The PCView device model is used to encapsulate a panel for showing the sensor and actuator states during the execution of the program. The other four device models correspond to the four NETMASTER controllers. It is seen that these four device models have similar structure. The resources named “MGR” are of type RMT_RES and are responsible for the deployment of the FBs contained in corresponding resource named “Control” which are of type EMB_RES.

![Fig. 5 IEC 61499 system model of the DCS in FBDK](image)

B. Development of the SIFBs

The distributed control system for controlling the didactic plant needed a number of communicating elements to establish communication between the controllers as well as to read from and write to the NETMASTER peripherals. Within the IEC 61499 compliant application the SIFBs represent these communicating entities. For maintaining the communication between the NETMASTER controllers the publisher and subscriber [9] SIFB pairs had been used. The SIFBs used for communicating with the peripherals of the NETMASTER are those described in [10].

Among the NETMASTER specific SIFBs only those for LCD, digital I/Os and keyboard were needed in this application. In addition to [10] FBs to make provision of extracting individual input bits and giving output values in the form of bits have been developed.
C. Development of the Control FBs

The different modules of the plant are responsible for different functionalities and control FBs are required to be constructed for each of the modules. These FBs should scan the necessary sensor inputs as well as receive the information from other modules that they require for their operation. It was identified that in each of the four stations one of the modules initiates the operation and then the other modules follow the operating sequence. For example, in the distribution station it is the feeder module that initiates and the transfer module along with its pneumatic suction cap follows the sequence. In the testing station it is the lifting module, in the processing station it is the rotary index table module and in the storing station it is the gantry crane module which plays the role of action initiator. Following these observations the above mentioned modules were assigned the role of deciding the action depending on the current state of the plant and then guide the other modules accordingly.

While building up the control FBs it was of major concern that the implementation of the FBs do not result in codes that occupy considerable amount of space as the NETMASTER modules, like most other soft-programmable controllers have memory constraints. Furthermore, the provision of including diagnostic facilities was somewhat overlooked since the traces of the deployed program running on the NETMASTER module helped us in this aspect. Moreover, test runs with a PC where the PCView (cf. Fig. 5) device model can be activated helps in validating the control algorithms.

In general the interface of the control FBs for different modules are similar. The only differing factor is the number of input and output variables. This is because of the varying number of sensors involved with the action of the modules and also because of the varying need of external signals from other modules. Fig. 6 shows the FB for controlling the feeder module. The interfaces of the control FBs for the other modules should differ only at the number of sensor inputs designated in the form “S_#” and outputs leading to the actuators written in the form “A_#” and additional input values from other stations or modules are denoted “E_#”.

![Fig. 6 FB for controlling the feeder module](image)

The input events RUN_MOD and INIT_MOD trigger the normal mode and initialization mode of operation. The input event SENSE is triggered by the digital input SIFBs when a change occurs in the corresponding NETMASTER controller’s input values resulting from the change of sensor values. The feeder module’s controller FB does not need any input signal from other modules, rather after a complete sequence of normal operation the RUN_MOD is triggered by the storing station’s crane module. After completing its normal mode or initializing mode operation successfully the control FB fires the RUN_CNF or INIT_CNF output events respectively along with the output qualifier QO and status string STATUS.

During the operation the actuator signals are activated through the ACT_O event firing along with the variables that holds the actuator values. This event is connected to the SIFB responsible for digital output activation and thus can achieve the execution of the desired action.

The ECC of the feeder module’s control FB is shown in Fig. 7. It shows that the triggering of the INIT_MOD event leads to a waiting state WAIT_INIT where it waits till the sensor inputs arrive. The sensor inputs are then scanned and accordingly actions are taken in ACT_INIT, if no changes are activated the initializing mode actions are confirmed to be finished successfully. Otherwise, the FB waits for the sensor changes due to the action and then decides whether the operation ended successful or not. Only after initialization the FB can move to normal mode of operation and after successful completion of the normal mode action the RUN_READY state is restored which implies that the FB is ready to operate in normal mode in the next sequence of operation.

In case of more complex control FBs the initializing or normal mode operation consists of several ECC states. Each of the states then activates a particular sequence of actuator signals depending on inputs, waits for the fulfillment of the action, and goes to the next state. In building up the ECC states the SIPN description proved to be quite handy.

![Fig. 7 ECC of FEEDER_CONTROL FB](image)

D. Implementation of the Control Algorithm

The most advantageous aspect of the migration from SIPN-based PLC program to an IEC 61499 compliant DCS was that considerable amount of work concerning the development of the control algorithm and its validation was partly circumvented. The SIPN-based design consisted of both sequential and concurrent algorithms. During the migration the sequential algorithms have been used to develop the ECCs for individual modules while the
concurrent algorithms have been achieved through the distribution of the tasks to FBs that can be triggered concurrently.

E. Deployment of the FBs

As shown in Fig. 5, each of the device models contains a resource named MGR which is of type RMT_RES. These resources in coherence with FBRT deploy the distributed application that run on different NETMASTER modules. Since the control application is totally dependent upon the communication and coordination of the NETMASTER module the PC where the application was developed can be disconnected without any interruption of the operation of the plant. The NETMASTER modules can be accessed through telnet and ftp and thus any modification can be loaded and the application can be launched with the modification. However, prior to the re-launch, a previous application needs to be stalled along with the operation of the plant due to implementation reasons (Java Virtual Machine).

F. Event-driven versus Cyclic Control

The application described so far is an event-driven one, though an attempt to develop a cyclic control system also preceded. In the cyclic implementation, SIFBs are triggered cyclically to read the sensor inputs and accordingly the actions are decided and activated through appropriate SIFBs. The stochastic nature of the Ethernet interface that leads to unknown delays in communication, as well as the lack of knowledge on the execution cycles of the NETMASTER module forced us to set a relatively high cycle-time (10ms) in our implementation. The plant under consideration had sensor signal impulses of varying lengths and the shortest impulse was of about 4ms duration. This obviously does not satisfy the requirement, cycle-time ≤ shortest sensor impulse and hence we had to go for the event-driven solution.

If it were possible to impose a cycle time shorter than or equal to the shortest sensor impulse duration, then it would have been better to implement a cyclic controller rather than an event-driven one. This is due to the fact that the indeterminate delays through the Ethernet may cause improper sequencing of the events, resulting in unprecedented action triggers or erroneous states. To avoid the effects of this improper sequencing of events in our event-driven implementation we circumvented passing such sequenced sensor impulses through Ethernet by passing signals denoting occurrences of a particular sequence.

Moreover, some of the actions need to be followed promptly after the occurrence of an event; in such cases cyclic controllers may often prove inadequate. For example, the rotary indexing table’s motion needs to be controlled such that as soon as the inductive sensor has sensed that the table has arrived any of the positions which are 90° apart from each other the motor should be stopped. Cyclic control at this point is quite demanding of precise and accurate time bounds, which were not achievable using the current set-up. In contrast, event-driven control reacts on the momentary changes of the sensor values immediately and therefore, satisfied the constraints better.

V. CONCLUSIONS AND OUTLOOK

In this paper the observations and experiences gathered while migrating from a PLC control to an IEC 61499 compliant DCS is shared. The didactic plant for which the control system was developed was a modular one and therefore the distribution of the control application was not quite troublesome. Moreover, the formal specification of the PLC program written in SIPN also eased up the control algorithm development.

One important experience gained during the work was that, when a non-deterministic communication medium is used cyclic controllers are preferable as long as the controller device’s execution cycles are comparatively shorter than the shortest sensor impulse so that the cycle time can be kept less than or equal to the shortest impulse length. Otherwise, it is prudent to develop event-triggered control for IEC 61499 compliant DCS ensuring proper sequencing of event signal transfers through the communication medium.

The SIPN based development was such that for the control of modules that perform identical activities, particular framework of SIPN was reused. With respect to this reuse as well as the hierarchical development concept SIPN and FB based developments are quite similar. Though within the scope of current work the translation from SIPN based algorithm to the IEC 61499 compliant one was done manually in further works an automated conversion is planned.

REFERENCES