Abstract - Functionality based Control is an approach that improves the quality of the design process in Distributed Controlled Systems (DCSs) by combining Object-Oriented concepts with re-configurability on function level. The basic functions are implemented in function blocks and then the overall functionality is derived by the use of a supervising scheduler. This paper presents an approach to model the basic functionalities in manufacturing systems using Unified Modeling Language (UML). A transformation of the UML description into an IEC 61499 compliant format is given. The approach is illustrated using the functionality based control (FBC) of a laboratory plant.

Index Terms – IEC 61499, Design Reuse, UML, DCS

I. INTRODUCTION

Current development of control software in manufacturing systems faces the problem of the escalation of complexity. The development process should provide short project duration and low cost at the same time. Therefore, the concepts for a control strategy shall provide high flexibility in the designing process. To solve this problem, the improvement of the design quality gained through reusable software and hardware modules seems to be a good solution [1]. In distributed Systems the Function Block (FB) concept is currently adopted and widely used by control engineers. It firstly was initiated in the IEC 61131 standard for Programmable Logic Controllers (PLCs) and later was extended in IEC 61499 for Distributed Control Systems (DCSs). IEC 61499 solves the some problems in the earlier concept related to the distributed nature of the new systems and also already acknowledges basic benefits of Object-Oriented (OO) technology. This new paradigm ultimately summoned up the development in the field of Industrial Process Measurement and Control Systems (IPMCSs). Although it gives an important direction for next generation manufacturing systems, the provided models inside this standard are not sufficient to describe distributed control applications throughout the design process. Therefore, as underlined in [2], existing mature methodologies, Engineering Support Systems and development frameworks should be considered to enhance the quality of the design process.

Meanwhile, a number of issues are rising to offer possible solutions dealing with flexibility and re-configurability in manufacturing systems. Design enhancement is exploited as one of the prominent items in effort to give a better comprehension for developers with different backgrounds. The requirements of the effective design are then pointed out, i.e. re-configurability, reusability, and understandability. Hence, the most preferable design approach along with its benefits is still open to be rectified in the field of DCSs.

Coming from those necessities, the functionality based control (FBC) approach tries to give a way out for an effective design process [3]. This concept concisely proposes functional modules based on the generic mechatronic functions in the context of automation processes since similar processes in different kinds of components are often realized with different software modules. In addition, a task scheduling strategy is also proposed in relation to the re-configuration of process activities. The scheduler as control manager operates the selected functionalities – implemented by the constituent function blocks – by means of pre-defined operation sequences [4]. This approach gives re-configurability in terms of the ability to repeatedly revise and rearrange the design of a system in a cost effective way. On a higher level, the next requirement is to find a suitable tool to design a system implementing the FBC approach again considering the reusability of the created artifacts.

In system engineering, the Unified Modeling Language (UML) seems to be an adequate notation for such demands. Many researchers from the object oriented community already engaged themselves to establish the usability of UML diagrams for FB-based system. This paper will address a cost-effective design in the sense that the designer should not do repetitive works in system design especially not for a new scenario of the processes which actually may contain a similar description. In short, a generic/re-usable FBC component model using UML is proposed. Here, genericity describes the level of generality of a solution or an approach in the context of basic mechatronic process in manufacturing systems. It means, the more generic the more multifaceted the reuse of the solution or an approach could be.

The rest of this paper is organized as follows. Section II describes briefly about DCSs based on the IEC 61499. Section III explains the FBC concept and its control strategy using a supervising scheduler. Section IV, the design of FBC components modeled using UML is discussed, followed by a description of the transformation to the IEC 61499 FB concept in Section V. Thereafter, Section VI will illustrate the design process using a didactic plant from FESTO. Finally, Section VII draws some conclusions and lights up an outlook.
II. DISTRIBUTED CONTROL SYSTEMS USING IEC 61499

In recent industrial automation there is growing interest in new technologies and architectures for building the next generation of distributed automation systems. This fact grows up since traditional DCSs and centralized control system using Programmable Logic Controllers (PLCs) tend to be difficult to modify and extend. They do not provide the high degree of flexibility that will be expected in systems for advanced and flexible automation [5]. Recently, the IEC (International Electrotechnical Commission) has developed the IEC 61499 standard for those reasons. This standard defines how the Function Block (FB) concept can be used to design and implement software for distributed Industrial Process Measurement and Control Systems (IPMCSs) [6].

The FB as functional unit of software is built by two parts, i.e. a head and a body. Events will flow on the head part and data on the body part. To describe the internal behavior of the basic FB instances, the Execution Control Chart (ECC) is provided. This chart helps the programmer by means of decomposing a complex behaviour into smaller pieces called states, where each of the states is only valid under a certain set of conditions. The states are further associated with one or more algorithms and output events. The activation of an EC state implies the execution of the attached algorithms and the consequent firing of its output events. Additionally, there is some resemblance to current OO-technology in IEC 61499. Encapsulation along with the reusability of function block types is one of the salient features that the standard inherits from the OO-paradigm.

III. FUNCTIONALITY BASED CONTROL (FBC)

As a new concept in automation systems, many works introduce design approaches based on IEC 61499 intensively. In [7] design pattern was exploited by J.H. Christensen as a pioneer in building a framework for IEC 61499 to simplify some of the unfamiliar concepts in IEC 61499 to the practitioner. Existing patterns namely proxy, Model-View-Controller (MVC), and distributed application have been restated. Afterward the extension of MVC and a layered MVC architecture was proposed in [8] with an example using FBDK tool. Model-Driven Architecture (MDA), another concept from object technology, has been applied by Heverhagen et al. [9] and Thramboulidis [10] extended in [11]. Besides, work in [12] investigated the reusability in design aspects regarding IEC 61499. However, to improve the reusability for implementation and modeling aspects a solution is still missing.

The Functionality Based Control (FBC) approach is proposed by extending the benefits of previous concepts in terms of design process and exploits reusability and re-configurability in function level. The functionality itself stands for a particular process/work corresponding to the devices in an automation system. The reason behind using FBC is that the mechatronic devices in automation processes with identical tasks are frequently conceived by using miscellaneous control software. Consequently, the system design takes much more time and energy in creating new functions. Therefore, generic functional blocks should be built at first. For instance, a function “move 2 points” can be employed to different mechatronic components such as forward-reverse conveyor, pusher, and up-down movement of driller. Later, the designer can use and reuse the available generic functional software and its models connecting them to get a control system.

Furthermore, to manage all activities of the FBC modules by means of interconnections and executions of the selected function blocks a task-scheduling approach is proposed. This approach is used to manage the execution schedule of functionality units along with their operation modes such as setup, normal, and emergency stop which can be implied as single task and/or multi-task. For the implementation a generic structure of Scheduler, Selector and Synchronizer (S³) is proposed. The architecture of the relation between S³ and FBC can be seen in Fig. 1. The arrows with solid lines show the event flow and those with dashed lines depict the data flow. The process can be described as follows: The Scheduler that gets control data from internal or external sources will order the Selector to run a schedule consisting of one or more functionalities. The Synchronizer will monitor the completion of the functionalities. After all tasks are completed, the Synchronizer gives a confirmation to the Scheduler and requests the next schedule. In case there is an interruption from internal or external systems or one of the functionalities takes longer than a pre-defined time assigned by the Scheduler, the resume process recorded by the Synchronizer will be confirmed to the Scheduler. Then, the process will resume the old task when a new start-up order is given. The combination of FBC and S³ contributes a design methodology firstly by means of encapsulation of control logic with generic mechatronic components (hidden intellectual properties) and secondly through reconfiguration in terms of new schedules at run-time which is possible even on low-cost and/or slow (weak computational power) hardware.

![Fig. 1 Architecture of FBC and S³ Task Scheduling Strategy](image-url)
IV. FBC MODEL USING UML

Because IEC 61499 does not give a sufficiently transparent model in terms of behavior and design process, the effort in this research attempts to find out an effective way for those features. The trend using UML diagrams came up as they are capable to support different features at various stages of development process providing the developer a flexible way of specifying the underlying concept. Works in [3], [4], [7], [8], [9], [10] and [11] used UML to describe their design approaches. The growing interest and the capability of such design techniques motivate the authors to use UML Diagrams as modeling tool for designing FBC components and their controllers. Reference models according to IEC 61499 are also considered as a guide to illustrate the proposed approach.

In Annex C of IEC 61499-1 [6] a description of function blocks using UML class diagrams has been already detailed. Hence, in this paper the explanation will focus on their reusability in automation processes. FBC components will be modeled in function block level.

Some class diagrams based on the FBC concept are shown in Fig. 2. They display examples of generic functionalities for mechatronic components: Move_2_C (interruptible), Move_2_Set (uninterruptible), Action_1_C (with time action), Move_1_T (with initial transition) and FIFO_memory. These class diagrams can be associated with the other diagrams by using aggregation relationships, for instance Move_2_C has relationships with Move_2, E_CYCLE and E_DELAY. Aggregation is a special type of association that implies logical or physical ownership. To define the kinds of data, event, and type of class, stereotypes are used to extend the core of the semantic. For example, input and output variables are expressed by using <<input>> and <<output>> stereotypes respectively. <<event_input>> and <<event_output>> represent the input and output events. In the same diagram, the indication of the reference level of the class diagram is shown with <<FBType>>. All these classes can later be instantiated into objects in order to augment reusability in the level of implementation and modeling. The level of generality of the classes determines the reusability in design process.

For describing the behavior of the basic level in each class of a certain class, UML state chart is employed. A state chart is a behavior diagram expressing a state machine as defined by David Harel [14]. State machines are useful because they provide a means of decomposing a complex behavior into smaller pieces, where each of the pieces is being only valid under a certain set of conditions. To illustrate the way to describe the behavior of the FBC component, Move_2 using a state chart is sketched in Fig. 3. This chart is a one-to-one mapping of the Execution Control Chart (ECC) adopted by IEC 61499 that is defined as an execution control section of the function block type declaration, considered to reside in the head portion of the block [6]. It consists of EC State, EC Transition, and EC Action. EC State represents particular state depicted by using <<ECState>>. EC Transition represents a combination of the required input event and conditions as a guard. The EC Action, contains two elements, first an algorithm linked to the invoked state and second an event output that will confirm completion of the algorithmic process and update the output data. It is identified by stereotype <<ECAction>>. Other FBC-components such as Action_1, FIFO_memory, and Move_1_B have been described by using the same kind of diagram.

The next design step is in composite components level. According to IEC 61499, input/output events in a composite function block can be interconnected with the input/output events to characterize the sequence and causality of function block invocations. The detailed rule of this function block level can be seen in [6]. Its class diagrams with generalization relationships describe the static model and a deployment diagram reveals the object interconnections. Firstly, in the static models shown in Fig. 2, Move_2_SET, Move_2_C, Action_1, Move_1, and Move_1_T are composite classes and they aggregate the other related classes. For example,
Move_2_C assembles some classes, i.e., Move_2 as control algorithm, E_DELAY as a time constraint for the execution of this FBC modules and E_CYCLE that used as updating sensor information for the affiliated algorithm. Secondly, the object interconnections are presented by using component diagram as a variant of deployment diagram. The interconnections should also show the relation of point-to-point connections among objects. Fig. 4 delineates an example of this diagram for Move_2_C. The Interconnection of the objects is divided into two parts, data connection and event connection using <<DataConnections>> and <<EventConnections>> stereotypes. For example, event Start from outside will invoke the event Start in the Move2 and the related algorithm inside this object will be executed and then confirmed by an event StartO that is linked to CYCLE_SENSE and DEADLINES. The data associated with the particular event will be transferred as well.

V. UML-FB IEC 61499 TRANSFORMATION

It has to be noticed that the transformation in this area implies the operation which maps one configuration or expression into another in accordance with a particular rule. The work in [13] is also related to this effort using stereotype identification to differ the level of class diagram regarding transformation purpose. In this section, the transformation concept will be presented as intuitive translation since the implementation of the automatic transformation is still ongoing work.

Annex A in [6] which details the standard of Document Type Definitions (DTDs) lists some recommended types: Data, FBType, Adapter, Subapplication, Resource, Device and System. Some of these items will be focused in this section. Those DTDs which are defined in eXtensible Markup Language (XML) specification [16] are used for the exchange of IEC 61499 library elements between software tools. To get the same meaning, the data structure in the UML diagram should be translated into IEC 61499 compliant XML. In the coding phase, the translation process will be divided into two categories, System (consisting of devices and resources) and FBType (consisting of BasicFB, ServiceInterfaceFB, and CompositeFB). The algorithm below illustrates a simple rule for translation in the system stage:

0] Get data structure from UML Diagrams (XMI,OLE,...)
[1] Data mining (eliminating the unimportant data)
[2] Select SYSTEM to get system data structure:
  [2.1] write xml version and address of DOCTYPE
  [2.2] write system name, i:=0
  [2.3] Read Device(i),
      IF Device=end THEN go to [3]
      ELSE
      BEGIN
      [2.3.1] Write Device(i) name, j:=0
      [2.3.2] Read Resource(i,j),
          IF Resource=end THEN go to [2.3]
          ELSE
          BEGIN
          [2.3.2.1] Write Resource(i,j)
          name,etc
          + FB (name, parameter,etc)
          + EventConnections
              IF FBNetwork!=end THEN
                  back to [2.3.2.2]
          ELSE
              IF Resource!=end THEN
                  (j:=j+1),back to [2.3.2]
              ELSE
                  (i:=i+1),back to [2.3]
          END (i:=0, j:=0)
          [3] END (i:=0, j:=0)

In order to get the particular kind of information, the reading process will search for the related stereotypes. For example, the system transformation process from an UML diagram to SYSTEM in IEC 61499 will search the <<system>> and the process translation (writing) will be done in the root of this system stereotypes. For the lower level models, such as Device, Resource, and FBType, the rules are almost similar dependant upon the characteristic of data structure on each reference.

VI. EXAMPLE

The example process consists of two stations in a didactic plant from FESTO (Fig. 5): Station_1 (Distribution) and Station_2 (Testing). In this process two FBC components are used: MOVE_2_SET and Action_1. The functionalities and their representations in the mechatronic components of the given system are summarized in Table 1. It can be seen that the reusability of the generic components based on FBC is improved with mechatronic components in the automation system. However, further functionalities like service interfaces for communication and I/Os are needed for implementation. For their description sequence diagrams are used.

Table 1 FBC components for Station_1 and Station_2

<table>
<thead>
<tr>
<th>FBC Components</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOVE_2_SET</td>
<td>Feeder</td>
</tr>
<tr>
<td></td>
<td>Lifter</td>
</tr>
<tr>
<td></td>
<td>Transporter</td>
</tr>
<tr>
<td></td>
<td>Pusher</td>
</tr>
<tr>
<td>Action_1</td>
<td>Suction In</td>
</tr>
<tr>
<td></td>
<td>Suction Out</td>
</tr>
</tbody>
</table>
To describe the scenario of the system, use cases are employed. It is a named capability of a structural entity in a model. They define the capability on a system-level without revealing or implying any particular implementation of them. Fig. 6 illustrates the use cases for the previous example named supplier system (Station_1 and Station_2). The three important cases for the operator in this system are noticed. Operation modes are used to determine a particular category which should run. Interruption is used to stop the system temporarily when the operator gives an emergency-stop event. The last case is Pre-definition of the schedule of the task executions and their time constraints to ensure that the tasks will be finished in time otherwise a process recovery should be done.

Furthermore, the relations among objects are revealed by using UML Sequence Diagrams. In this work, such diagrams are used for scenarios to get knowledge about the system not used for implementation. Some process synopsis on the subject of operation modes can be implemented with some sequence diagrams. For example, the scenario of normal mode with the accepted work-piece of the supplier system is manifested in Fig. 7. The reusable class, i.e., Move2 and Action1, is instantiated in FBC component objects as constituents of control process as well as the task manager using S³ for both stations. Communication interfaces as a media to share information, even the event-driven ones across the resource and/or the device level is simplified by encapsulating into the communication object. In implementation, PUBLISH_n and SUBSCRIBE_n component will be used respectively to send and to receive the data and/or events. SERVER_m_n and CLIENT_n_m have also been rendered pertaining to communication purpose in IEC 61499. It is insecure in this example to both stations at the same time. Therefore the sequence between lifter and transport should be synchronized. The process will start the normal operation from station 1 distributing a work-piece to test place (station 2). Initially, event Start will be send to S³_Station1 as a supervising scheduler of station 1 via communication layer. It manages all activities applied using four automation objects, i.e., Feeder, Transport, Suction_in, and Suction_out. The first two are instantiated from the generic class Move2 and the later two from Action1. For finalization one or more object instance sequences are established by sending event END to the S³_Station1. When all processes have finished, the information will be sent to Station 2 to start the testing process controlled by supervising scheduler (S³_Station2).
The other scenarios can be described in the same way. Besides that, other diagrams which have close relations to sequence diagrams could also be used to describe the dynamic behaviour. For example, collaboration/interaction diagrams show the behaviour in terms of object relationships. They convey the same information as sequence diagrams but they focus to object roles instead of times that messages are sent.

To describe all the components at system level, the use of deployment diagrams using nodes is appropriate. The deployment diagram for the example of the supplier system (station 1 and station 2) is shown in Fig. 8. It gives a picture of the relations among objects and components in the system level. Some objects linked to the existing FBC components will be aggregated to FBC resources regarding their own station. Then this group will be interconnected to the other object like S³ and service interface input/output to build a bigger component named resource. This resource component is linked again to other resources to assemble an IEC 61499 compliant device level description. Finally, the device will be communicating to each others in system level using service interface to interchange the data and drive the event among them.

Fig. 8 Deployment Diagram for Supplier System

VII. CONCLUSIONS AND OUTLOOK

In this paper an approach to improve reusability in IEC 61499 regarding the Functionality Based Control (FBC) has been explicated using UML diagrams. The proposed approach introduces a framework to assist system designers to build their manufacturing automation system in a cost-effective way. The use of UML diagrams to describe common functionalities of mechatronic objects has been illustrated at an example for better comprehension. The control system is generated by instantiation from the built UML-classes and appropriate interconnection of the functions (scheduling).

A concept for re-configurability at function level has been explained as well. In this concept, a new scenario of process functions is implemented by changing the operation schedule using the scheduler, selector and synchronizer (S³).

Moreover, a transformation concept has been briefly presented using XML as media for data interchange between data structures in UML diagrams and an IEC 61499 compliant XML.

The next step in the presented work is the realization of an automatic transformation from UML to IEC 61499. The result will not only help to avoid errors in manual translation but also lead to combine the design levels of the system development.

REFERENCES