FUNCTIONAL CONTROL OBJECTS IN DISTRIBUTED AUTOMATION SYSTEMS

Seno Panjaitan and Georg Frey

University of Kaiserslautern, FB EIT, JPA²
Erwin-Schroedinger-Str. 12, D-67653 Kaiserslautern, Germany
{panjaitan | frey}@eit.uni-kl.de

Abstract: Flexibility and re-configurability are the current paradigms which should be considered in manufacturing systems. To deal with such capabilities, the systems are considered as assembly of intelligent components (mechatronic objects) settled in a distributed control framework. In this paper, Functionality Based Control (FBC) is exploited to enhance flexibility dealing with high reusability of common functional mechanisms in mechatronic objects. The simplification of some activities such as re-designing and re-programming on component level is an advantage of the approach. FBC incorporates a scheduling mechanism for basic functionalities. This could not only enhance the reuse but also supports the cost and time efficient re-configuration of the system to new production scenarios. While object oriented concepts (namely UML) are used in the design, the implementation of the presented structures is using the current industrial standard for distributed control systems: IEC 61499. Copyright © 2007 IFAC

Keywords: distributed automation, IEC 61499, mechatronic object, functional module, intelligent component.

1. INTRODUCTION

Distributed manufacturing systems began to be a trend by the end of the 20th century. They can be defined as manufacturing systems with constituents that are geographically distributed and operated in a cooperative way in order to work as a whole. A Distributed Control System (DCS) is a subset of the distributed manufacturing system, responsible for controlling and coordinating the various constituents (machines or work-cells). At the beginning, the traditional DCSs were using a few large central processors to provide supervisory control and data acquisition to the sub-ordinate units in a strictly hierarchical setting. However, such architectures tend to be difficult to modify and to extend and do not provide the high degree of flexibility that is expected in systems for advanced and flexible automation. Hence, a new concept or generation of DCSs is needed. This will be a process automation system where the control is organized by co-operation of intelligent mechatronic devices.

In the area of DCSs, IEC 61499 as a new programming framework has been standardized. This technology concept is supported by some industries and many researchers to be a preferred solution for the future manufacturing technology. O³NEIDA is a community participated by industrials and academicians which support this standard and also proposed an open, object-oriented knowledge economy for intelligent industrial automation as an initiative for enabling decentralized, reconfigurable industrial control and automation in discrete manufacturing and continuous process systems (Vyatkin et al. 2005).

The concept of mechatronic object is currently introduced in the automation field and becomes a vital part of the automation object concept. It is based on an object-oriented (OO) approach in the context of the mechatronic perspective. Mechatronic devices (modeled as mechatronic objects) are vital parts in the automation system corresponding directly to the real environment as controlled objects. Some previous works considering mechatronic objects are based on centralized automation structures. However, in distributed systems reusability is a prominent capability to deal with in order to reduce cost and time of development life cycle. The level of intelligence incorporated in a mechatronic object determines its flexibility and re-configurability on higher levels of control. However, the more components are used the more connections/relations among components have to be handled. This issue will be discussed in this paper and a solution to it will be proposed. It concisely exploits the reusability of mechanism func-
tionality on mechatronic devices as common functional objects in collaboration with the design and implementation of component modules. Mechatronic devices then can instantiate one functional object as a basic control or compound several objects (i.e., composite module) if a component employs several functionalities. Functionality Based Control (FBC) is then proposed to improve reusability of the functionalities. FBC also offers a possible way to add flexibility and re-configurability to the design. Together the presented concepts provide a fundamental basic control scheme for mechatronic objects as a part of automation components and their requirement elicitation.

The rest of this paper is organized as follows. The next section gives an overview of selected works regarding OO ideas in automation. Section 3 describes component dimensions and Section 4 explains the proposed component based approach. The elicitation of components is then illustrated in Section 5. Section 6 gives an example and Section 7 provides a summary and an outlook.

2. MECHATRONIC AND OO-IDEA

In the automation area, there are considerable efforts to adopt ideas from OO technology. This phenomenon has been influenced by the need of flexibility and re-configurability. Mechatronic object is one of the terms which try to bridge the knowledge gap between OO-idea and the automation perspective. Mechatronic device is a primary component in process automation and object is a key part in OO idea. The profitable combination seems to be a good option by taking the advantages of each of them.

The initial work combining the mechatronic and object-oriented term can be seen in (Bonfe and Fantuzzi, 2003). Here, the mechatronic system compounds mechatronic objects with signal-based interfaces. A mechatronic object, e.g., function block, should have (1) an interface for signal and data, (2) an input/output interface to the process (private), a private data set to store status information and a control algorithm. Concisely, one mechatronic object takes responsibility for one mechatronic device. An example for the use of IEC 61131-3 and IEC 61499 function block (FB) was depicted as well in the context of this work. In other works, a mechatronic component in hardware part, named Physical Mechatronic component, includes two parts: physical functional and embedded control device (Vyatkin, 2003). A Virtual Mechatronic Component has also been discussed in (Sündor, et al., 2005). Work in (Thramboulidis, 2005) also focuses on the system development with mechatronic component in low level and mechatronic system in the higher level. It was mentioned that primitive mechatronic components are developed with generic functionality. However, how to provide the specifications of generic functionality such as operation mode and development concern of generalization activity was not detailed yet.

In the mechatronic object idea, the logic control of mechatronic devices which may have one or more functionalities are composed in an object. The obstacle in improving the reusability and flexibility may appear when there are many mechatronic devices with different logic control but still provide similar functionality. Here, re-programming and re-design has to be done for all control algorithms. However, it can be augmented by providing the common functional object to minimize the cost and time.

Another idea is Automation Object (AO). It is defined as an abstraction of a mechanical device associated with its embedded intelligence, i.e., software components of different functional domains. The proposal for AO concept can be seen in (IEC TC 65, 2002) where the reference model is defined by means of UML and the transfer syntax is defined using XML. Its feasibility in the application of automation system has also been exploited conceptually by OOONEIDA (Vyatkin, et al., 2005) by proposing the architecture of AO. The internal dependencies in an AO, e.g. component layout data relates to the appearance on its visualization, will give a leeway to alter the component automatically if their relations are described formally. The benefits of Intellectual Property (IP), i.e., encapsulation and reuse, can further be brought by using such intelligent components and object-oriented design. IEC 61499 has been noticed to become a cornerstone for development of AO concept. For example, a FB can be used as a basic control of an AO and its diagnoses then can be done by capturing the data from input/output interface. Compared to mechatronic object idea, AO is more general. It could be said the mechatronic object is a subset of AO. However, currently there are still some needs for more clarity of AO concept in order to give consistency among developers, vendors and users. A standard to guide the usability of AO is also required.

3. DIMENSIONS OF THE COMPONENT

The essential dimensions of functional module (component level) can be classified into composite/basic entity, abstraction/concretization, and generalization/specialization in the context of model, FB-source code and hardware. These dimensions (cf. Fig.1) will be boundaries and terms concerning the functional object/component development.

![Fig. 1. The dimensions of component development.](image)
**Composite/Basic Entity** dimension represents the characteristic of component functionality which may or may not compound particular entities. In the model, a composite model compounds one or more models herein. In FB or source code, one or more function blocks/source codes as well as their interactions are joined inside of a composite one. In the hardware level, an application module may consist of one or more mechatronic devices working cooperatively. The activities around this dimension could be divided in two ways, i.e., decomposition and composition. **Decomposition** is an activity to subdivide a module into smaller parts that are easier to understand and control. **Composition** represents the contrary activity where some components are put together along with their interconnections into a composite module.

**Abstraction/Concretization** dimension shows the relation of the component concerning the implementation of the system. The abstraction is a view of a component as an object that focuses on the information relevant to the particular purpose and ignores the rest. The concretization is the area in real environment of implementation such as physical layer or hardware. Two activities in this dimension, which move in an opposite way, are verification-and-validation and embodiment. The first activity is aimed to validate and verify the concerned information represented by models that humans can understand and analyze. The target is to check the correctness of the model/source code to guarantee and provide confidence with the design. The embodiment activity in opposite way moves toward more and more to executable representations.

**Genericity/Specialization** dimension represents the concern of component regarding its reusability or in contrast may be only used for a special purpose. Genericity is a capability of component to be generic where the model, source code or hardware can be reused on building a new component. Specialization represents a unique component which has less reusability and may instantiate a generic component. **Generalization** and **instantiation** are the activities in this dimension. Generalization is the activity which tries to make a component as generic as possible. The concern of component development should be considered as the limitation of its genericity. In contrast, the instantiation is the activity to reuse the available component, which is more generic, in order to build a new component.

4. FUNCTIONALITY BASED CONTROL (FBC)

In accordance with control strategy on component or mechatronic object, the migration from the conventional way to the proposed one can be seen in Fig. 2. The control strategies in the conventional way normally are based on the process interfaces of a mechatronic object, i.e., list of sensors and actuators, where the properties of the mechatronic object itself are not important. It means that the core functionality in this approach is in the logic part which contains all control strategies. The reusability and re-configurability on such way are however hard to be extended. The change of format, functionality and other dynamic properties make it quite difficult. Only in case that the whole logic will be re-used then it will be useful.

The proposed approach considers the control logic as hidden element of a mechatronic device. Interface of sensors and actuators will be replaced with the list of basic functionalities. Such functionalities will compose their intellectual property herein and as a component they compound not only their functional programming but also their behavior models. In addition, the rules for combination of such functionalities (schedulability point of view) are taken into account for components integration. A task scheduling concept containing the components Scheduler, Selector and Synchronizer (S³) is applied to deal with components integration and to add re-configurability for the execution of components. As a result, a designer or programmer only selects the component functions (FBC approach) and provides their schedules of operation to the Scheduler in S³. Unavailable component functions can be either included in the repository where the component is stored as a new common basic functionality or built by interconnecting the available components. Otherwise, they can be included in the repository as new components.

The FBC approach extends the benefits of previous concept, i.e., mechatronic object and automation object. The functionality itself stands for a particular process corresponding to the automation devices. It embeds intellectual property to improve the genericity of functional object used to control the mechatronic objects such as move 2 points, move 1 point, action 1 (action without sensor and use time constraining its process), FIFO memory, etc.

For example, a basic common functionality named move 2 points has the following properties:

- Two target points: home and end position,
- Two kinds of action: e.g. forward and reverse, up and down, etc.,
- Interruptible or uninterruptible option for an action,
- Direct (e.g. forward and reverse afterwards are done using one event trigger) or indirect movement (e.g. one event to trigger forward action and
then another event to trigger reverse action),
- It has time constraint, e.g. WCET (worst-case execution time),
- It considers some operation modes, e.g. normal, emergency stop, etc.

The generic specifications of move 2 points can be reused for several components as shown in Fig. 3 (instantiation activity). Semi-rotary actuator, which is used to handle a work-piece gripped by a suction component, moves it from home position to the end. For conveyors, there are some kinds of conveyor with one or two sensors indication. In the figure, it can be seen that roller conveyor with two sensors can be controlled as well by using function block move2 representing the move 2 points functionality. The position of home and end is depending on the requirement in installation. Pick-and-place and transporter specimen/work-piece employ the similar functionality to semi-rotary actuator moving on vertical or horizontal direction. Furthermore, vertical movement of gripper also can be controlled by using move2 where home is position on the top and end is position on the bottom (on top of piece).

Furthermore, the use of UML leads to the reuse of the available abstraction model so that the behavior model of a functional object, e.g., move 2 points, can be reused for describing a component using the similar functionality. Besides, the basic functional object can be applied as a basic FB type, which embodies the related code, according to IEC 61499 as the current distributed application standard. A basic FB type can compose an Execution Control Chart (ECC) as internal behavioral model of its execution. The instantiation of the FB type reuses all of its properties hence the reusability of FB type is enhanced if it provides a more generic functionality (generalization in the context of source code). The remaining task that has to be done by the designer or programmed with a generic functional object is the configuration of its external variables such as input/output interfaces and employed events. Previous work related to the design of components employing UML tools can be seen in (Panjaitan and Frey, 2005) and (Panjaitan and Frey, 2006). Both papers show another benefit using UML where another stage in development life cycle such as requirement artifacts also can be described in a proper way.

5. ELICITATION OF THE FUNCTIONALITIES

Functional elicitation here is defined as an activity to capture the requirement of functional control objects. As discussed before, some common objects can be built and put in the repository to be used and reused for the development process. Therefore a requirement on a functional module can be done by analyzing and deciding which functionality in the repository is compatible with it. In case that the functionality of the device control does not exist in the repository, a new functional module (generic or special) has to be built. A mapping from mechatronic device to the functional requirements employing set relation theory can be seen in Fig. 4 as a digraph. The set theory provides a simple way which could be adopted easily by any actors in development process. The sets of the digraph shown in Fig. 4 are detailed as follows:

\[
D = \{D_1, D_2, D_3, D_4\}; F = \{F_1, F_2, F_3, F_4\}
\]

The relation between set of D and F can be written as:

\[
R: D \rightarrow F
\]

Where:
- R : a relation definition between two sets, i.e., D and F.
- D : a set of mechatronic devices
- F : a set of related functional requirements, in repository or as a new demand(s).

Another representation using roster notation for the example of Fig. 4 is given below:

\[
R = \{(D_1, F_1), (D_1, F_3), (D_2, F_1), (D_2, F_2), (D_3, F_3), (D_4, F_4)\}
\]
A composition of functionalities in a mechatronic object is done such that D1 requires functional F1 and F3. Likewise, device D2 demands F1 and F3. The composition will reduce the scale of representation in the model. It is possible because some relationships between the two functionalities are hidden on the level of mechatronic objects relations.

The instantiation of functional source furthermore is an important part in relation to the reusability. In Fig. 4, the instantiation of F1 for both D1 and D2 as well as F3 for both D1 and D3 are examples that functionalities can be reused for some mechatronic objects. The degree of their reusability is higher than F2 and F4 in this example. FB/source, for example, D1, which instantiates the F1 is not allowed to be named similarly. Hence, it is named D1.F1 (in the example) using the dot-notation from the OO domain.

It has to be noticed that the digraph in Fig. 4 and the roster notation only focus on the requirement interpretation in the sense of used functionalities. They do not include the required sequence of operations. For the definition of a sequence of functionalities from F mapped by the Relation R onto the mechanical devices D, the following notation can be used:

\[ S = \langle D1.F1, D1.F3, D2.F1, D2.F2, D3.F3, D4.F4 \rangle \]

In each sequence, the functional elements with the corresponding device (using the dot-notation) are arranged in an ordered list separated by commas and enclosed by corner brackets. Depending on the device and its functions the concurrent execution of functionalities may be possible and required. In this case the symbol "||" is used to combine two functions concurrently. (e.g. \( \langle D1.F1 \mid D2.F2, D3.F3, \ldots \rangle \))

Often the elements are represented by names to increase readability. To do so, a mapping from the functionalities of the system onto unique identifiers has to be defined as follows:

\[ \{D1.F1 \rightarrow push, D1.F3 \rightarrow move, D2.F1 \rightarrow pull, D2.F2 \rightarrow count, D3.F3 \rightarrow convey, D4.F4 \rightarrow drill\} \]

With this definition the above example results in:

\[ S = \{push, move, pull, count, convey, drill\} \]

The sequence can be changed according to changing demands on the manufacturing system or to describe different functional model of the devices.

Furthermore, for a manufacturing system, there are often several possible sequences of operation which should be handled (e.g., associated with different products to be produced). Some operations between them can be defined such as concatenation, filtering and so on. For instance, the sequences \( \langle P_1, P_2, P_3, \ldots, P_n \rangle \) and \( \langle Q_1, Q_2, Q_3, \ldots, Q_m \rangle \) are given. The concatenation between these sequences can be represented as \( \langle P_1, \ldots, P_n, Q_1, \ldots, Q_m \rangle \). This kind of function represents FIFO (First In First Out) behavior in a system where the first sequence should be finished and followed by the second one.

The filtering between two sequences can be represented as follows:

\[ \langle P_1, P_2, P_3, \ldots, P_n \rangle \text{\{Q_1, Q_2, Q_3, \ldots, Q_m\}} \]

For example,

\[ \langle convey, drill, convey, test, convey \rangle \text{\{convey, drill\}} \]

yields the sequence \( \langle convey, drill, convey, convey \rangle \).

Filtering is used when a system only employs particular functionalities while in an incoming sequence there are one or more unknown functionalities. It produces a new sequence possible regarding the system. Some other operations like intertwining can be included dependent upon the system requirements.

The sequence can then be combined with time constraints for each task producing a set of task schedules. Later, these schedules of operations will be put as input to the scheduler module as an important part of a system or sub-system.

6. EXAMPLE

Fig. 5 shows a Festo plant regarding the distributed control application as a didactic Modular Production System. Sketch of the system can be seen in Fig. 6. Goals of the system are to scrutinize the cylindrical work-piece for thickness and material type, drill a proper hole into each piece, and then sort them according to their material types. In case the piece is not drilled properly then it should be rejected.
Fig. 6. Sketch of the modular production system.

For the system, there are four stations employed so called supplying station, testing station, processing station, and sorting station (cf. Fig. 6). The mechatronic components in all stations will apply the following generic functionalities: Move 2 points (F1), Move 1 point (F2), Action 1 (F3), and FIFO memory (F4). If the functionalities are not provided in the repository yet, then they have to be built in the initial development.

The requirements elicitation of the employed mechatronic devices are described as follows:

- **Supplying station (WS1):** Push/pull cylinder (D11), Transporter (D12), Suction in (D13), and Suction out (D14).
- **Testing station (WS2):** Lifter (D21), Push/pull cylinder (D22), FIFO memory (D23).
- **Processing station (WS3):** Conveyor (D31), Rotary table (D32), Drill module (D33) composing clamper (D331), driller motor (D332), and up/down cylinder (D333), and Tester (D34).
- **Sorting station (WS4):** Transfer module (D41) composing up/down cylinder (D411), suction in (D412), and suction out (D413), Transfer forward-reverse (D42), and Transfer left-right (D43).

The relation of functional object requirements can be represented by using roster notation as follows:

\[ \text{R}_{\text{WS1}} = \{D_{11}, F_1, D_{12}, F_1, D_{13}, F_3, D_{14}, F_3\} \]
\[ \text{R}_{\text{WS2}} = \{D_{21}, F_1, D_{22}, F_1, D_{23}, F_4\} \]
\[ \text{R}_{\text{WS3}} = \{D_{31}, F_2, D_{32}, F_2, D_{331}, F_1, D_{332}, F_3, D_{333}, F_1\} \]
\[ \text{R}_{\text{WS4}} = \{D_{411}, F_1, D_{412}, F_3, D_{413}, F_3, D_{42}, F_1, D_{43}, F_1\} \]

Furthermore, the requirement of operation schedule of each station can be described as follows:

\[ \text{S}_{\text{WS1}} = \langle D_{11}, F_1, D_{12}, F_1, D_{11}, F_1 | D_{13}, F_3, D_{12}, F_1, D_{11}, F_1 | D_{14}, F_3 \rangle. \]
\[ \text{S}_{\text{WS2}} = \langle D_{21}, F_1 \rangle \ \text{IF piece=good THEN} \langle D_{22}, F_1 | D_{23}, F_4, D_{21}, F_1 \rangle \ \text{ELSE} \langle D_{21}, F_1, D_{22}, F_1 \rangle. \]
\[ \text{S}_{\text{WS3}} = \langle D_{31}, F_2, D_{32}, F_2, D_{331}, F_1, D_{332}, F_3, D_{333}, F_1 \rangle \]
\[ \text{S}_{\text{WS4}} = \langle D_{411}, F_1, D_{412}, F_3, D_{413}, F_3, D_{42}, F_1 | D_{43}, F_1, D_{411}, F_1, D_{413}, F_3, D_{411}, F_1, D_{42}, F_1 | D_{43}, F_1 \rangle. \]

Later, the coordination among stations can be done by the higher level schedule among them. For instance, the simple coordination schedule is \(\text{S}_{\text{WS}(1-4)} = \langle \text{S}_{\text{WS1}}, \text{S}_{\text{WS2}}, \text{S}_{\text{WS3}}, \text{S}_{\text{WS4}} \rangle\). The reusability values of common functional objects in the four stations are \(F_1 = 9, F_2 = 2, F_3 = 4,\) and \(F_4 = 1\).

### 7. CONCLUSION AND OUTLOOK

An approach for the design of distributed controllers exploiting the fact that mechatronic objects often show common functional mechanisms has been elucidated in this paper. The logic control of a mechatronic device is replaced by a higher logic scheduling only the pre-programmed appropriate functionalities. An illustration how to choose the functionalities for the employed mechatronic device has also been illustrated. Future work will concentrate on verification of the functional relations in system level and standardization of the manufacturing functionalities.

### REFERENCES


IEC TC 65 (2002), Automation objects for industrial-process, measurement, and control system, Working Draft of IEC TC 65: IPMCS.

Panjaitan, S. and G. Frey (2005), Functional design for IEC 61499 distributed control systems using UML activity diagrams, Int. Conf. on Instrumentation, Communications and Information Technology (ICICI’05), Bandung, pp. 64-70.


