Combination of UML Modeling and the IEC 61499 Function Block Concept for the Development of Distributed Automation Systems

Seno Panjaitan and Georg Frey
University of Kaiserslautern, FB EIT, JPA²
Erwin-Schroedinger-Str. 12 D-67653
Kaiserslautern, Germany
e-mail: (panjaitan,frey)@eit.uni-kl.de

Abstract

This paper proposes Unified Modeling Language (UML) as modeling tool for distributed control systems (DCSs) according to the IEC 61499 standard. The reason behind using this model is a demand to describe the system considering the whole development process. The Object-Oriented (OO) concept in UML along with its supporting tools is a good candidate for this purpose. However, IEC 61499 is built on the use of interconnected Function Blocks (FBs) as a modeling paradigm. A mapping concept from UML into FB-based systems and its implementation are presented. As a result, the UML model using class diagrams, packages, and state diagrams can be transformed automatically to models according to IEC 61499. The proposed process leads to a combination of previously disjoint design levels in the system development.

1. Introduction

Today, system development tends to be more complex due to the effect of dynamically changing market demands. Consequently, the design process should not just consider a particular step in the system development such as software modeling, but also include the whole scheme to deal with more complete description. For that reason, the Unified Modeling Language (UML) seems to be a suitable tool since it provides various diagrams to cope with different steps in the development process and allows building models on different levels of abstraction. In the automation domain, the IEC 61499 [1] has proposed some reference models to be the framework in Industrial Process Measurement and Control Systems (IPMCSs) regarding distributed control systems (DCSs). It retains the use of previous programming methods in basic components built on the – by now accepted – standard IEC 61131 for the programming of Programmable Logic Controllers (PLCs). One goal surely was to allow an easy migration from the previous concept, in order to make the new one more acceptable by practitioners in the field of automation. The main concept of IEC 61499 is to build a system by the interconnection of Function Blocks (FBs). These FBs share information by event-based communication mechanisms and encapsulate their private data and algorithms. For the description of the algorithms themselves the languages of IEC 61131 are proposed. Therefore, IEC 61499 provides good concepts for the (fine-grained) design and the implementation of a DCS. However it lacks in support for the other steps in the development process.

In the beginning steps of system development such as requirement analysis and first (high-level) design decisions as well as in the analysis of the system structure and functionality other modeling tools are required. In the software engineering domain, UML is the predominant choice for these tasks today. UML can be used to cover the tasks that are not covered by IEC 61499. Furthermore, it can bridge the gap between software design and automation engineering, if a transformation of the results from UML to IEC 61499 is provided. As a result, two alternatives are open: The first alternative is completely modeling the system using UML tools and then automatically transforming it to an FB environment by producing code according to some data-interchange standard of the FB runtime system. The second alternative will use UML only to describe aspects not well covered in IEC 61499 and generate a partial model to be completed using FB tools like FBDK [2] or CORFU [3] (e.g. structure generated in UML with algorithms designed in the FB environment). In both cases, the aim is to give better comprehensibility not just in programming level, but also in the different steps of system development.

The rest of this paper is organized as follows. Section 2 will present some reasons why UML should be used in combination with IEC 61499 and how it is already used in other approaches. Section 3 details the concept and the software implementation for the transformation followed by Section 4 which illustrates the result using an example. Finally, Section 5 will conclude the paper and give an outlook on further work.
2. Overview on UML and IEC 61499

2.1. OO Modeling Concept in Distributed Control Systems

The object-oriented concept originated in computer science and it is currently an influential paradigm in software development. It brings up some effects in automation system since many processing units of automation controllers are based on software programming. As a result, some ideas in this technology start to be adopted although in this time the adoption of the whole OO paradigm is still expensive to be implemented in the field of automation systems. Therefore, by economical and practical reasons some primary concepts regarding modularity, reusability and interoperability as well as the improvement of behavior description including the ability to analyze the system development are concentrated on. Nevertheless, the current efforts in this direction convey the OO-concept into distributed automation systems and pose a major challenge in software development.

In fact, there has been steady growth in the functionality provided by control systems due to advances in both hardware and software in the last 50 years. The OO-idea is believed to be influential in the future for the development of industrial control technology [4].

Based on this motivation, IEC 61499 sews some of the OO-ideas with existing concepts from the automation domain (Function Blocks) to achieve comprehensibility and acceptability of users with various backgrounds. However, in deciding on a given level of accepting OO concepts a trade-off between keeping familiar concepts and introducing modern techniques was faced. Hence, although IEC 61499 gives an important direction for next generation automation systems, the provided models documented in the current standard are not yet sufficient to completely describe the design process.

Finally, this use of OO techniques will impact the development approach for automation systems. UML itself does not describe a development process. By using UML, developers are free to employ many existing design approaches such as “4+1” architecture [5], Component based approach [6], Waterfall model or V-Model [7] and so on.

2.2. IEC 61499

In distributed systems, the FB paradigm is widely used by automation engineers and intensively discussed by academics and industry practitioners. The concept was already used in the IEC 61131 standard for Programmable Logic Controllers (PLCs) and extended in IEC 61499 for distributed systems. The new standard also acknowledges the basic benefits of OO technology in this domain.

In the modeling technique, IEC 61499 provides the FB-based design concept. Based on the IEC 61499-1, there are some reference models such as System, Device, Resource, Application, Function Block (basic, composite, service interface, adapter and proxy), distribution, management, and operational state model. The Execution Control Chart (ECC) within an FB will select the state that is active. Then, this state will run the algorithm associated with it. The communication and interconnections among FBs adopt the event-driven approach where the signal condition of an event input at an FB may depend upon event outputs at other FBs. To get more details about the models please refer to the IEC 61499 standard document [1].

In the implementation step the use of IEC 61499 could be sufficient but it still faces some limitations in the scope of design and requirement analysis. Existing design approaches directly based on IEC 61499 are mainly based on the adaptation of existing design patterns such as the well-known Model/View/Controller (MVC) and Proxy restated firstly by J.H. Christensen as a pioneer of the development framework of IEC 61499 [8]. Later this concept was extended in [9]. Other approaches, for example Functionality based Control [10], already employ additional tools to cope with the limited modeling power of FBs.

The need to elevate the model and analysis tools to have more capability of underpinning various aspects at various development stages is clear to see. Such tools should offer the developer a leeway of specifying the development concepts in a better way. It was found that the number of modeling tools and the system aspects description are more comprehensive in UML than in IEC 61499 in terms of system development.

2.3. Combination of UML and IEC 61499

The applicability of UML in the sense of automation engineering however still provides inadequate notation. Reasons for this are pointed out in [11], i.e. the lack of a guideline or strategy, the UML standard needs to concern with this area. The advantage of inheritance is considered to be weak in this area. Furthermore, specific tool support, automatic code generation for automation systems, and practical experience using UML in this area are still missing. Therefore, the combination of UML with concepts originated in automation such as IEC 61499 will improve its applicability there.

Furthermore, there are some reasons for the use of UML in IEC 61499 modeling documented from the experience of some researchers. In [12] it is stated that in the analysis phase IEC 61499 does not define the way that requirements will be captured and formalized neither the way that these requirements will be transformed to design specification. In [13] it is avowed that despite IEC 61499 provides numerous advantages, it still does not use all the potential benefits of the object-oriented and component based concepts. Other works aiming to bring together the UML and
FB-based engineering methodologies express similar reasons (see [14], [15]). In the presented work, some more specific arguments are enumerated by comparing the provided models in UML and reference model in IEC 61499 (see next sub-section).

For the translation of UML artifacts to FB environments a translation based on the use of UML class diagrams is proposed in [13]. To represent the hierarchy of FBs, aggregation relationships are used in the so-called UML-FB class diagram and to depict the intra-level and/or inter-level connection between FBs, association relationships are employed. Each class diagram has a stereotype along with its attributes (FB’s variables) and operations (FB’s event). By using this information, the transformation is done by producing the XML-representation of the resulting FB-system. In a basic block, the event input as an operation will execute a related algorithm which should have the same name. The implementation is done using Rational ROSE. Another transformation approach using a model-driven approach for integrating UML notation into the FB environment has been proposed in [16] using the CORFU-FBDK Engineering Support System.

In the course of this paper an additional alternative tool to transform UML into FBs is given. The use of class diagrams, packages and state diagrams is exploited and the data/event connections are delineated by using connection interfaces where the condition of some particular data and events at one FB may depend on data and events at other FBs. Besides, the logic inside of a states associated algorithm can be shown clearly with sub-state or activity diagram.

### 2.4. Comparison of UML and IEC 61499

The comparison is divided by two kinds of modeling scope: function-modules and overall-system.

In function-module scope, shown in Figure 1, the behavior model in IEC 61499 is limited to the FB level. The state machine named ECC is applied only in basic FBs. Likewise, time-sequence diagram is a modeling tool employed only for Service Interface FBs to describe the sequences of communication with the other FBs. The FBs’ interconnection can be depicted with composite function block. On the other side, the provided UML diagrams can delineate in more detail the behavior inside of the algorithms linked to the states (e.g. using Activity Diagrams) associated with the states of a State Diagram used for the same purpose as ECC in IEC 61499. As a result, the logical description of the algorithms can be modeled in UML in more detail.

In the overall-system scope which defines how the model should be realized, the comparison between UML and IEC 61499 is pointed out in Figure 2. In this scope there could be relationships between high-level and lower-level of the components. IEC 61499 furnishes some reference models for building a system.

The interaction between models as interconnections among FBs can be seen under this model. In the implementation level the parameters concerning the execution parameters and network protocols of the system can be configured. This model can be enriched by a multitude of diagrams in UML to manifest the requirements from the developer with characterization of some scenario cases which can be revealed in usage model and even more with activity diagram to describe the process flow in the system level. Furthermore, the interaction among components can be modeled using sequence diagrams and/or interaction diagrams.

![Figure 1. Comparison of IEC 61499 and UML in the scope of function modules.](image1)

![Figure 2. Comparison of IEC 61499 and UML in the scope of overall-system.](image2)

In the translation step, not all of those models can be used. Hence, there are two kinds of data: description artifacts and transformation data source. All of them will be used for modeling but only the latter is taken for conversion purpose. It has to be mentioned that there are approaches that try to use description data for other tasks in the development process. For example information annotated to sequence diagrams can be used for the automatic generation of test cases for the implemented system [17].
3. UML-FB Translation

Some reasons, as mentioned previously, motivate the authors to build an automatic converter from UML diagrams into IEC 61499 models. It will bridge the gap between designer backgrounds. Accordingly, the automation designer can still directly use an IEC 61499 tool for implementation and on the other hand the software designers which are already familiar with UML can design the system using specific UML diagrams described below in the analysis and design phase, and then transform them into a FB system.

3.1. Function-Module-Scope (FB-level)

This sub-section explains the mapping rule which is implemented in the presented work. There are two main parts on a FB, i.e. external interface and internal part. The external interface is used to connect an FB to other FBs. Such a connection is described using an interface element and its counter-part, which depends on the information from it, will be linked using a dependency relation. This relationship element signifies that a single or a set of model elements requires other model elements for their specification or implementation [18]. Figure 3 shows the mapping of UML component diagram and class diagrams into the external part of a FB which is operated as connection interface. The event signal is indicated as <<signal>> stereotype and input/output(s) associated with it (in IEC 61499 known as WITH relation) are depicted as attributes of the signal. For instance, variable IN and OUT are associated with event EI and EO respectively. The type of those variables is specified in the parentheses.

Figure 3. Component and Class diagram for an IEC 61499 FB’s External Interface.

The behavior of a basic FB, which is described by using ECC in IEC 61499, can be depicted by UML state diagram (SD) as well. One-to-one mapping is done for this case as shown in Figure 4. The transition in SD has two meanings, i.e., guard and/or action. Guard (left side) is used to represent an input-event and its associated variable(s) and action (right side) is used to publish an output-event after an algorithm in the associated (pre-)state has been finished. If there is no event and variable in guard, it will be denoted as 1 (based on IEC 61499 semantic). Moreover, the initial state in ECC will be described in SD with the state that gets the initial pseudo state. It means when this SD is called, the process will start from that state. The algorithm in a state will be associated by putting an action on the state. For instance, do/ALG_MAIN in Figure 4 means that if the MAIN state is invoked then the algorithm ALG_MAIN will be executed.

Figure 4. Modeling of an IEC 61499 ECC using an UML State Diagram.

For the algorithm itself IEC 61499 adopted some programming languages from IEC 61131-3 such as Structured Text, Instruction List, Function Block Diagram, and Ladder Diagram. Beside that, another option using JAVA is allowed too. However, in all these cases, the algorithms are treated somewhat separately from the rest of the model. For that reason, an additional – graphical – model is helpful to transparently specify and understand the program logic inside a basic FB. The UML activity diagram can be used for this purpose. The formalism is closely related to Petri nets and allows a comprehensive graphical description of sequential and concurrent algorithms. Figure 5 shows an example of an algorithm description for ALG_MAIN associated with the state MAIN in Figure 5 modeled by an activity diagram.

In addition to basic FBs, IEC 61499 defines composite FBs. A composite FB is built by grouping an interconnected set of basic or composite FBs. Interconnection here means the separate connection of event and data inputs and outputs. Figure 6 shows the mapping in terms of composite FB. In UML, a package can compose some components inside. Similarly, such concept is also adopted by composite FB. The important thing here is the data/event(s) flows. Even though they have been already described, but it seems that it is focused on an interconnection relationship distinguishing the data and the event. It needs more explanation by means of sequences. Therefore, to analyze more deeply such behavior further UML diagrams could be used. The interaction between components
can be shown for example using sequence and/or interaction diagrams. By using those kinds of diagram, the time sequence of operation and/or tasks can be described properly. For the external interfaces of composite FBs, the concept is similar to the basic FBs. The event may or may not be associated with particular external variables (input/output). This kind of relation is described in the interface element in UML diagrams. The interface itself is a connection object. By implementing this kind of object, the constraint or the other connection rule among diagrams can be specified as clear as possible.

Figure 5. Algorithm description using UML Activity diagram.

Figure 6. Mapping of Composite FB.

3.2. Overall-System-Scope (System-level)

The explanation of the mapping rule from UML to FBs has been elucidated above. The next higher level in IEC 61499 is the resource model where FBs will be interconnected and then realized to be constituents of the process control system. The resource model actually contains interconnected FBs similar to a composite FB but without explicit external part since the connection among resources is realized by service interface FBs as communication interface. Furthermore, the resources communicate among each other through the higher (device) level. Finally, the connected devices build a system. In IEC 61499 the specific behavior concerning time-sequences on this level is not defined. As a remedy, UML provides the diagrams for describing some aspects in this level. For the description of sequences, in resource level as well as in composite FB level, UML sequence diagram can be used. As an example, Figure 7 illustrates the mapping of interconnection between components in the resource level.

Figure 7. Mapping of interconnected FBs in the Resource level.

Already for this simple example, it can be seen that the sequence of execution is not described clearly such as which event should be executed first, which event will respond and confirm the signal, and so on. To deal with this requirement, the sequence diagram in Figure 8 will help to understand the process queue controlled by the FB’s interconnections. The composite sequence can also be described like in the frame sd FBC_1 for composite FB named FBC_1 and in the frame sd Resource1 for the resource itself. As the result, such model will improve the comprehensibility and testability in order to avoid errors in connecting the FBs.

For more details about the concept of the resource level and above (device, system) please refer to [10] where this mapping was described in the framework of the functionality based control approach and to the example in Section IV which will illustrate these levels in more detail. The Device and Resource could be implemented as component package where all components regarding resource and device will be sketched herein.
Finally, in an actual development process normally at the very beginning, there should be a usage model. However, there is no concept for a usage model in FB-based systems according to IEC 61499. In UML, the use case diagram can be employed to show the general scenarios of the system. Figure 9 depicts an example of the use case diagram for a scenario of the system. It shows that the process should be started by an operator (human) and the operation will start the selected mode depending on the information which is given by the operator from the system mode into operation mode module of supplier system. Then the monitoring will work by using the value/data sent by the supplier system. Furthermore an interruption could be issued by the operator too.

Figure 9. Example of an UML Use Case Diagram showing a process scenario.

The mapping as mentioned before is based on the component paradigm. Each component can contain some structural and behavioral descriptions. In addition, a component which composites or embeds other components inside with interactions among them can be described using interaction diagrams. However, at this time in the implementation of the automatic transformation, there are only several diagrams that will be taken into account such as component as a subtype of class diagram, packages, and state diagram.

3.3. Software Implementation

In this work, the Function Block Development Kit (FBDK) from Holobloc Inc. [2] is used to view the Function Block as a result of the transformation. Real time Studio Professional from ARTiSAN software tool [19] is used as UML modeling tool. This software provides Object Linking and Embedding (OLE) automation which works by obtaining a root of an object. For instance, the project object of an ARTiSAN Real-time studio model can be enquired for attributes and relationships. By using this way, the ARTiSAN model can be explored as a network of connected objects. For programming JAVA is used. On the IEC 61499 side a standardized exchange format based on eXtended Markup Language (XML) is available and used in this work.

The structural model using UML class diagram of the built software is shown in Figure 10. There are two important parts, Reader and Writer. Reader is used to get the desired information from the UML diagrams. In this work the ArtisanReader will take information from the repository data using the OLE interface of the Real time Studio tool. The obtained data is scanned to get the needed information that will be stored in the reader buffer to be used in conversion. Later, the data is used by the Translator to generate FBs and the other reference models of IEC 61499, i.e. Resource, Device and System. Furthermore, to run this application a class named CommandLineApplication is used. With this class, the user can choose which project model that will be converted and to which folder the result (function block type, resource, device, and system) will be saved. Finally, the result can be opened in FBDK and translated there for implementation with the corresponding runtime environment.

Figure 10. Class diagram for the UML to FB Transformation tool.
4. Example

To describe how a design in UML can be transformed into an FB-based model, a part of an automation process example namely the supplier station shown in Figure 11 is considered. This station has to supply work-pieces continually to the following station. In the normal operation mode a pneumatic cylinder pushes one work-piece on the pick area and then the transporter will move forward to take it with the suction. Then the cylinder moves back and stands by in home position. The process continues by lifting and putting that work-piece on the test place. Thereafter, the cylinder will supply the next piece.

![Figure 11. Supplier System.](Image)

In the following only those models are presented that are actually used in the automatic transformation. The system model includes some important modules such as operation mode, algorithm, and service interface. In the beginning, all those components are modeled in UML. The UML model as shown in Figure 12 describes the mentioned system where the hierarchy of all the elements is put on the left side which is also recorded in repository data as ‘dictionary’ which can be accessed by using the OLE Interface. In the main page of the project model, some diagrams are interconnected to each other to build the desired system design. Here, the composite, resource and device model are implemented by using packages with particular stereotypes as their identity to inform about which kind of reference model they represent. This identity is needed in the transformation process to know where the result of package translation should be located. The basic FB and service interface FB are shown as class diagrams. Their parameters can be set using a property menu. Stereotypes are also employed to distinguish the difference between the diagrams.

Besides, the relation between them can be embedded in composite FB and/or resource as shown in package CTRL_STATION_1. In this package there are two class diagrams, i.e. DELAY and ALGORITHM_
5. Conclusions and Outlook

The paper described the reasons for, the approach to, and the implementation of a transformation from UML to models according to the IEC 61499. The whole explanation addressed the use of UML in system design for distributed control system application by underlining the advantages such as the provided comprehensibility and extensibility which give the opportunity to bridge the gap between designers from automation engineering and software engineers.

The presented tool currently only implements the kind of reference model of IEC 61499 which is commonly used, i.e. not the complete model. Further work will be focused on improvements on this point. The analysis of the UML model under the special constraints of IEC 61499 will be an important subject too.

Acknowledgment

The authors like to thank Christian Walke for the help with the implementation during his Diploma thesis and his advisor Andreas Rausch from the Department of Computer Science at the University of Kaiserlautern for the helpful discussions on the topic.

References

[19] ARTISAN Website: http://www.artisansw.com/