IEC 61499 Architecture for Distributed Automation: the “Glass Half Full” View

Alois Zoitl

Automation and Control Institute (ACIN), Vienna University of Technology, Gusshausstrasse 27-29 / E376, A-1040 Vienna, Austria, zoitl@acin.tuwien.ac.at

Valeriy Vyatkin

Department of Electrical and Computer Engineering, The University of Auckland, New Zealand 1142, v.vyatkin@auckland.ac.nz

Citation:

A. Zoitl, V. Vyatkin, “IEC 61499 Architecture for Distributed Automation: the ‘Glass Half Full’ View”, *IEEE Industrial Electronics Magazine*, 3(4), pp. 7-23, 2009, doi: 10.1109/MIE.2009.934789

IEC 61499 Architecture for Distributed Automation: the "Glass Half Full" View

I. Introduction

Control software is the main element in today's industrial automation systems for providing the correct and save operation of the automation process. Furthermore such requirements as flexibility, adaptability, or robustness, envisaged in the visionary study from the Iacocca Institute [1], largely increase the complexity of the control software. Therefore, new means for developing control software are necessary. With the new family of standards for distributed automation systems, called IEC 61499 [2], [3], and [4], the International Electrotechical Commission tried to fulfill these new and upcoming demands. The standard is available since 2005, and has attracted substantial research attention, resulted in many publications (see survey in [5]), and several reference implementations [6]. However, its industrial adoption is rather low. Main reasons of the slow adoption are unresolved semantic issues [7], [8], lack of clear application and development guidelines, and missing industrial grade implementation platforms [5].

The standard's text is hard to read and contains some ambiguities [7]. Furthermore only limited tutorial information on IEC 61499 is available. Just two books explaining the ideas of IEC 61499 were published to the date: [9] and [10], the first being a little outdated as it considers the first draft version of the standard. This led to many misconceptions of the standard's ideas and contradicting conclusions of the researchers on the role and place of the standard in future industrial automation. For that reason the majority of control device vendors and users (e.g., control engineers) still cannot appreciate the advantages of using IEC 61499.

This article attempts to bring some insight into the concepts and models of IEC 61499 by discussing and analyzing common misconceptions. Especially we focus on two issues: modeling of distribution and architecture-centric design, which have not got substantial attention of researchers so far. Our analysis points out future research directions for increasing the adoption rate of IEC 61499.

II. Background and Current State of IEC 61499

In the last 20 years the standardization and research efforts related to control software for industrial automation was focused on two main points: (1) Improve the software quality and reliability; and (2) Reduce the development time of industrial automation control applications by reusing developed control software elements across different automation projects, and also across control devices of different vendors. The latter issue is commonly referred to as *vendor independence*. IEC 61499 was developed to meet these requirements. However, in this task it has to compete with the well established predecessor standard IEC 61131-3 [11], which has brought some relevant solutions, or at least improved the situations for the users.

A. Unification and Modularization with IEC 61131-3

One of the main goals driving the IEC 61131-3 development was to unify the programming concepts for industrial control applications. At the time when the IEC 61131-3 development started, a great variety of different programming languages and concepts have been used in industry. The IEC achieved to reduce this variety down to five programming languages, some of which are textual, and some graphical. Furthermore it defined a common way of handling inputs and outputs, data types, and control programs. At present, nearly every control vendor supports the standard, at least partially. Therefore, engineers knowing IEC 61131-3 can now switch easier between devices of different control system vendors. However, because of vendor specific extensions or only partial support of IEC 61131-3, direct reuse of developed control software elements is not really possible.

For structuring control applications the *Function Block* (FB) concept has been introduced. Similar to integrated circuits in electrical circuit design, a FB encapsulates a certain functionality and can be connected to other FBs via its data inputs and outputs (see Figure 1a). In IEC 61131-3 a FB can contain just one algorithm, which may be written in any of the IEC 61131-3 programming languages. Because of this limitation, the FB concept of IEC 61131-3 is often compared to a module in procedural programming languages. However we must not neglect that a FB encapsulates not only one algorithm but also data, and that it can maintain its state between invocations. These are the features of object-oriented programming languages, which are more sophisticated in facilitating reuse. However, IEC 61131-3 does not support such object-oriented features as inheritance and the interface concept.

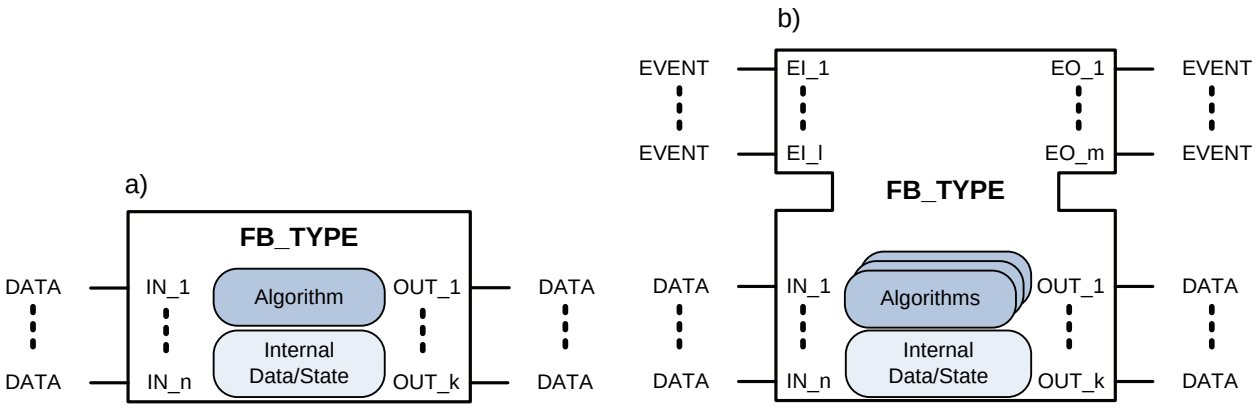

Fig. 1. The graphical representation of the Function Block concept in a) IEC 61131-3, and b) in IEC 61499

The encapsulation of application parts in FBs helps to modularize control applications and foster the reuse of application parts. However IEC 61131-3 has two main drawbacks hindering the reuse. First, it allows global data. Global data acts as a hidden interface between the FBs and makes seemingly separated FBs tightly connected. This leads to rigid program structures where parts may not behave the same without the remaining application part; and changes done locally in some application's parts may have unforeseen global consequences. The second drawback is that the application developer has no direct control on the execution order for the FBs in the application. In an IEC 61331-3 control system the execution order is derived from the connections between the FBs, according to the rules defined in IEC 61131-3 [11, pp. 249ff.]. These rules leave some interpretation room, therefore, the same application may work differently on different control devices.

B. Distribution and Component Orientation with IEC 61499

The mentioned weaknesses of IEC 61131-3 partially stem from the fact that the standard is now more than 15 years old, so its concepts are not state of the art in software engineering anymore. However, new software engineering trends cannot be taken directly into the industrial control systems domain. There are many specific requirements of the automation developers, and a typical control engineer has only limited knowledge in computer science and software development. The IEC took this into account for the development of the IEC 61499 architecture, which should support such new features of next generation industrial automation systems as

distribution and reconfiguration [1].

In order to leverage the existing know-how, the IEC 61499 architecture builds on top of the IEC 61131-3 definitions. The main element of the architecture is again the FB, but this concept has been extended in several ways to incorporate new developments from the domain of software engineering (Figure 1b). The most eye-catching extension is the event interface. A FB in IEC 61499 remains passive until triggered by an input event. On event, the FB executes and produces output events and data. The event interface has been criticized for making IEC 61499 applications more complicated compared to IEC 61131-3 applications because of additional event connections between FBs. However, in contrast to IEC 61131-3, where the FB execution order is implicitly determined by the development tool, the event interface in IEC 61499 allows explicit specification of the FBs execution sequence. This gives the developer a new level of flexibility not possible in IEC 61131-3.

Another potential problem created by event-triggered execution of FBs in IEC 61499 is the specification and implementation of real-time behavior. However, recent research results overcome also this limitation (see [12] for an overview on execution methods for IEC 61499 applications complying with real-time constraints).

There are a few other extensions of the FB-concept in IEC 61499 towards object-orientation and component-orientation. First, FBs may contain several algorithms similar to the methods encapsulated by an object. However, in contrast to an object, the algorithms inside a FB are neither visible nor directly accessible from the outside. Furthermore, there is no global data in IEC 61499. This greatly enhances the reusability of FBs as there are no implicit dependencies between application parts, so, removing or adding a FB influences only the connected FBs. Internal variables of a FB are also completely hidden. There are no means to access or change a FB's internal variable as it was possible in IEC 61131-3 with the access path mechanism.

Taking these properties into account, a FB in IEC 61499 is an independent software entity that can be implemented, tested, and used independently of other FBs. Therefore IEC 61499 much better supports the development and reuse of tested components (i.e., FBs), which will lead to better quality of industrial automation software.

C. Current Industrial Adoption of IEC 61499

IEC 61499 potentially brings many benefits for developing industrial automation systems. These were proven in numerous case studies conducted in academia and research institutes worldwide. However, the current adoption of IEC 61499 in the industry is still very limited. One of the first industrial automation engineering tools supporting IEC 61499 was ISaGRAF [13], a product of ICS Triplex (currently a part of Rockwell Automation). Since version 5 released in 2005, this IEC 61131-3 product has been enhanced with support of many of the IEC 61499 elements. It is now possible to develop distributed control applications in IEC 61499 together with application parts in IEC 61131-3. However not all concepts of IEC 61499 have been implemented. Thus, the communication between the application parts is achieved through network variables instead of *Service Interface FB*s (SIFB). FBs have an appearance according to IEC 61499, but their internals may look a bit different. For example, the *Execution Control Chart* of a Basic FB is defined by means of the IEC 61131-3 *Sequential Function Chart* language. An additional problem is that the event-driven IEC 61499 FBs are executed on top of a cyclic-scanned and time-triggered IEC 61131-3 runtime system, which results in an execution overhead and therefore in a performance drawback for the IEC 61499 application. A more detailed discussion on the specifics of ICS Triplex's implementation of IEC 61499 is given in [14]. Apart from the mentioned limitations ICS Triplex showed in several demonstrations consisting of up to 70 controllers the capabilities of their approach [15]. Several pilot systems in research organizations have been implemented using the IEC 61499 feature of ISaGRAF, but no real industrial deployment was reported so far to the best of the authors' knowledge.

Another example of the industrial scale IEC 61499 development is the Austrian company nxtControl that has developed an integrated SCADA and distributed control approach based on IEC 61499 [16]. They provide an industrial grade engineering environment which supports the design of control applications and visualization together in one tool. This approach has great advantages in productivity and reuse of both control and visualization components. The nxtControl engineering tool provides several features that have been long expected from IEC 61499, for example, a debugging and online-monitoring infrastructure, allowing to debug single FBs as well as fully distributed applications. Another feature is the automatic generation of the communication during the distribution process of the application. This greatly reduces the

engineering effort when distributed control applications are designed. nxtControl applied their technology already in eight projects in the domain of building automation. The largest project has been a training center building with 19 control devices controlling about 2500 I/Os (heating, ventilation, air-condition, lighting, etc.) with IEC 61499.

Several other companies are currently investigating how and when to move to or integrate IEC 61499 in their product lines. One reason of the moderate adoption pace is that the advantages of IEC 61499 over the established IEC 61131-3 seem only marginal, compared to the switching effort. Possibly, IEC 61499 may be beneficial only in some domains of industrial automation applications, and this requires further investigation and case studies. However, we see some exciting features of IEC 61499 which will help to achieve higher quality control applications and therefore deserve a closer look. In the following we will point out some of them that have not got attention they deserve.

III. Platform Independent Application Design

Design of distributed control systems is obviously more complex than of traditional centralized ones. To cope with that complexity, IEC 61499 offers modern platform independent approach to system design, similar to the Model Driven Architecture (MDA [17]) used in development of complex software and embedded systems. MDA consists of three main models allowing to develop applications in a generic model-driven way. These models are: the Platform Independent Model (PIM) for modeling the application; the Platform Definition Model (PDM) for modeling the target system (i.e., devices and the communication infrastructure), and the Platform Specific Model (PSM) which contains the assignment of the PIM elements to the devices and platform specific configurations and adaptations. The PSM is used to automatically generate the target specific code that will be executed in the devices. The MDA approach has greatly improved flexibility and efficiency of the development process for embedded systems [18] on account of re-using elements of the solutions, described in high level languages. The same solution can be easily implemented on a variety of targets, ranging from the code running on a standard hardware to fully customized hardware, implementing the same function.

The design process promoted by IEC 61499 is subdivided onto two steps. In the first step, the functionality of the whole system is defined using a PIM called *Application Model* [2, pp. 21ff]. This may require quite detailed specification of each function as a composition of

simpler functions. The Application Model is fully executable, however, it is free from particulars of hardware and communication protocols.

In the second step, details of a particular hardware configuration are taken into account in the form of a PDM called *System Model* [2, pp. 18f]. The functionality is said to be *mapped* to that hardware, resulting in the PSM called *Distribution Model* [2, p. 26].

We can expect similar benefits from IEC 61499 for industrial automation that MDA brought to software engineering and embedded system development. In the following subsections we will illustrate on a simple example the application development process of IEC 61499.

A. The Application Model

In IEC 61499, the platform independent modeling of control applications is done by instantiating and interconnecting FBs. A simplified example of such a control application is shown in the upper part of Fig. 2. The application implements closed-loop control. The application development can start from the closed-loop plant-controller model (Model 1 as shown in the left upper part of the Figure). The Plant FB can contain a simulation model of the object allowing for immediate simulation of the whole system (see [19] for more details of this design methodology). In the next design step the Plant model FB can be transformed to a composite FB with the same interface, but with direct references to sensors and actuators. This can be further transformed to the Model2 in the right upper part of the Figure. The FB `SENSOR` provides sensor data, this data is taken by the FB `Control`, which performs the control algorithm, whose output is passed to the actuator represented by the FB `Actuator`.

This example shows a further difference to IEC 61131-3: inputs and outputs (I/O) are not directly addressed in the application. Instead, interface to the I/Os is implemented in the form of FBs. This helps to isolate dependencies on particular I/O names and addresses, keeping the core function free of such dependencies.

The direct access to the readings of peripheral controller I/O modules is implemented by a special type of FBs called SIFB [2, pp. 43-54]. A library of SIFBs can be specific for a particular device hardware, but there are many possibilities to organize I/O access in a generic way, for example as shown above, using the “Model of Plant” FB, or a generic I/O FB with parameters specifying type and address. As an option, such a generic FB can implement also simulated I/Os that are set/observed from screen. In early design stages an application can be developed and

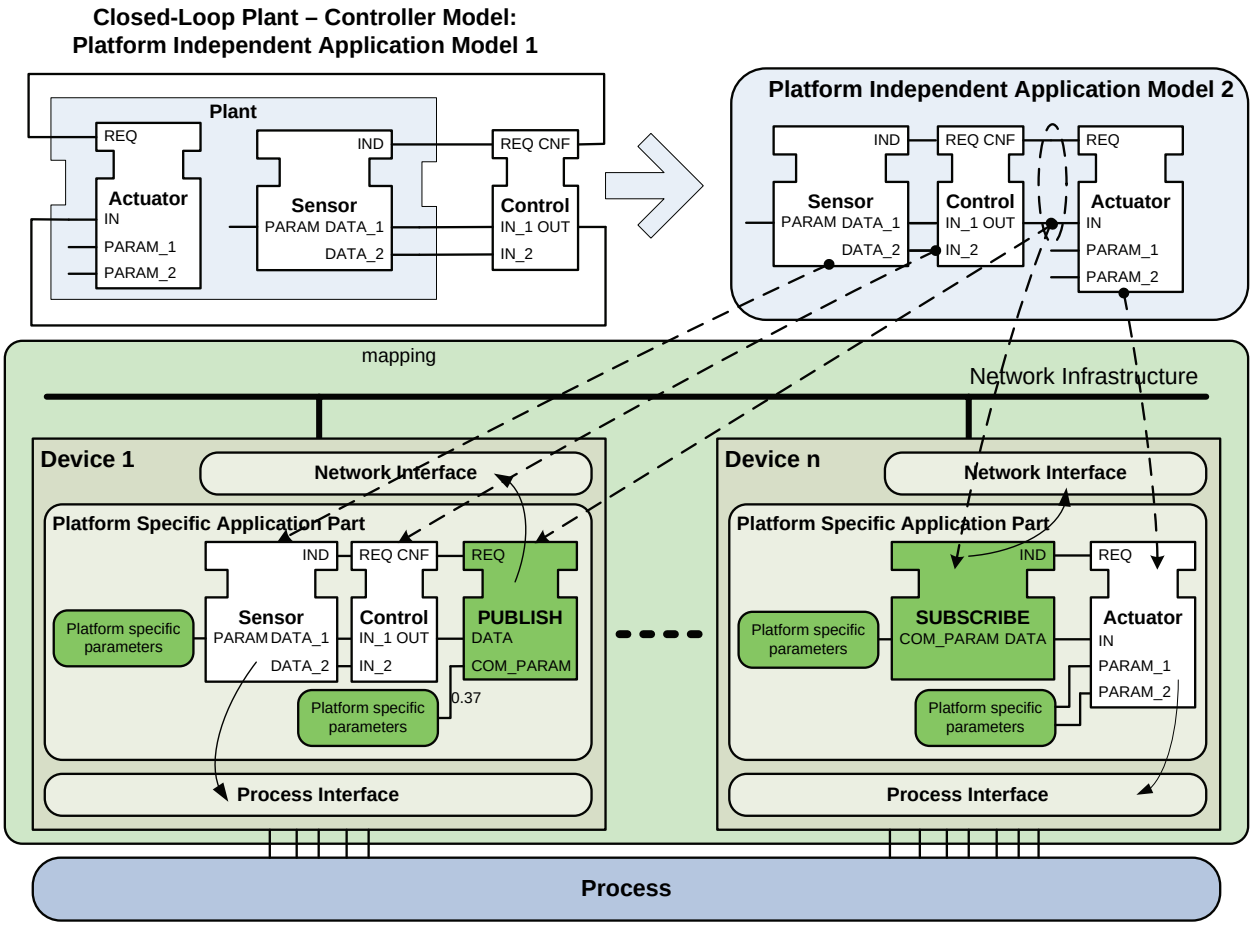

Fig. 2. Overview on the platform independent application model of IEC 61499 and its relation to the control devices in a distributed control system

debugged using such generic I/Os references. When deployed to a control device, parameters of these generic SIFBs can be changed to specify concrete addresses of I/Os modules.

The SIFB concept of IEC 61499 has been often criticized for its platform dependency. In our view, such critique is based on the incorrect assumption of using SIFBs with fixed addresses of I/Os from the early design stages.

B. The System Model

To execute an application on a distributed network of control devices, it is necessary to take into account particulars of the platforms composing the system. In IEC 61499 this can be done in the *System model*. Properties of control devices are described in IEC 61499 by the *Device*

Model [2, pp. 19f]. They contain a network interface for communicating with other devices, a process interface for interacting with the controlled process, and a library of FBs provided in the device (e.g., special SIFBs for I/O access). A device can contain *Resources* [2, pp. 20f] that provide an execution environment for applications. Resources can be regarded as independent *execution containers* for FB applications. They may provided specific functionality (e.g., special computation hardware), but may also just be used for logical separation of the device onto smaller independent entities.

The communication infrastructure is modeled with so called *Segments* and *Links* [2, pp. 18f]. A segment describes a network segment of a certain type (e.g., CAN) to which several devices are connected with links. One device can be connected to more than one segment (e.g., bridging or routing devices). This allows to freely model the hierarchical structure of the network topologies typically used in industrial automation.

In our example from Figure 2 we use just two devices: one connected to the sensors and one connected to the actuators of our control loop. We also have just one network segment connecting the two devices with each other.

An open issue of the IEC 61499's system model is that the descriptions of devices and segments are very generic. No specific parameters or methods are provided to describe needed communication or device parameters. Communication parameters would be needed for example to allow tools checking the available and used bandwidth. For device types, the main shortcoming is in the description of the communication and the process interface. There an extension towards specific ports would be needed in order to know which networks a device supports or which I/Os a device provides. In [20] a possible extension of IEC 61499's system model based on the *Field Device Markup Language* is presented. This work can be the basis for further investigations on needed extensions and improvements of the system model.

C. The Distribution Model

The final phase of the application development process in IEC 61499 is the distribution of the control application to the control devices. In this step the application's FBs will be mapped to the control devices where they will be executed on. In our example the `SENSOR` and the `CONTROL` FB are mapped to `Device 1`, and the `ACTUATOR` FB is mapped to `Device n`. This mapping is not just a logical link in the models; instead a copy of the mapped application

parts is created. The reason for this is that device specific changes may have to be done. Such changes are the specification of device specific parameters to the FBs. In our example we have to define the device specific parameters for the `SENSOR` and the `ACTUATOR` FB, so that they know which I/Os they have to provide.

As a result of the distribution, some FBs, connected to each other in the application via event and data connections, may reside in different devices. Such connections, going across device boundaries, need to be implemented by inserting communication SIFBs [2, pp. 47f]. In our example these are the `PUBLISH` and `SUBSCRIBE` FBs, added in the device specific applications. In more complex applications distributed across many devices the generation of the communication FBs can be rather burdening. However provided with the information from the application and system model, the design tool could automatically insert these FBs and provide suggestions for suitable communication parameters. The nxtControl Studio tool is the first to provide an implementation of this feature, which proven its efficiency in development of highly distributed systems. ISaGRAF inserts communication functions implicitly, making them hidden from the user.

In our sample application the mapping was done by copying the application parts to the devices. For simplicity we assumed that the generic SIFBs used in the application model are also supported in the control device. In a general case, the application will just specify with a generic SIFB that it needs, say, a digital input. When this generic "wish" is mapped to the device, the specific SIFB providing the digital input on this device has to be inserted, manually or by the tool.

IV. Architecture-centric Development with IEC 61499

Architecture-centric development is currently the dominating approach in the design of complex software systems. It improves quality and re-usability of the product. The software architecture should be defined in an early stage of the development phase and it should capture the requirements of the system. IEC 61499 has several means to capture architecture, application structure, and requirements already at early stages of the application development process. These means are described in the following subsections.

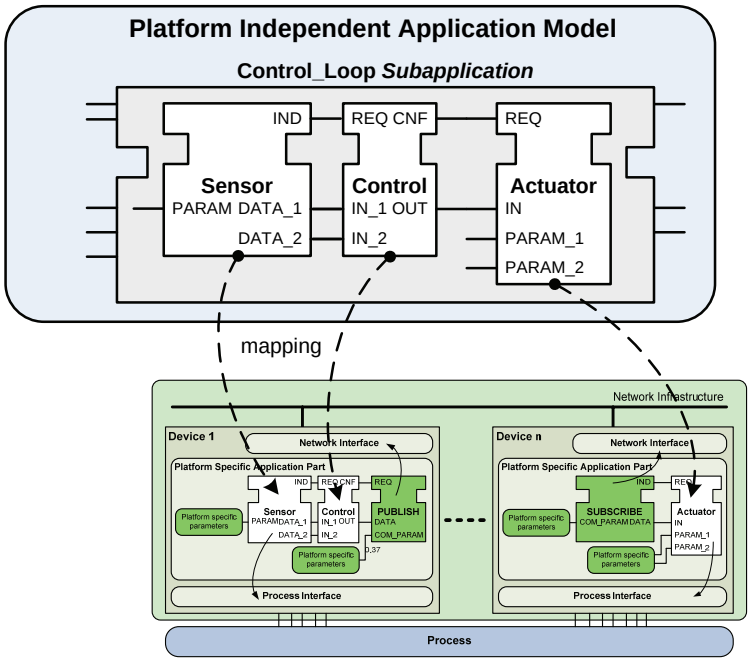

Fig. 3. The example closed loop control application encapsulated in a Subapplication

A. *Application Structuring with Subapplications*

A typical top-down application development process starts with specifying the top level application components and their interaction. In subsequent design steps these components are specified in more and more detail. The FB concept is only partly suited for such a top-down application development since a FB is atomic. That means a FB can be later assigned only to one device. However top-level or even medium level application components may encapsulate functions of several control devices (e.g., the control of a whole machine). Therefore other encapsulation artifacts are necessary that allow internals of such components to be distributed to different devices. IEC 61499 provides such a design artifact called *Subapplication* [2, pp. 37-39].

The top-down application development does not cover all the design scenarios, the bottom-up approach is also necessary. In the bottom-up approach, application parts can be grouped to subapplications. This can be illustrated on our closed-loop control example. We can encapsulate it in a subapplication providing the interface of the closed-loop control to higher levels (e.g., updating the set-point), while still being able to distribute the FBs to the two control devices (Fig. 3).

Currently only the open source engineering tool 4DIAC-IDE [21] provides some basic support of subapplications. As pointed out in [22], this tool will be further extended towards saving subapplications or loading subapplication templates as suggested by [23].

B. Typed Interfaces with Adapters

Another problem of IEC 61499-based design is clarity and readability of applications composed of many large FBs. Such FB diagrams usually have many data and event connections cluttering the design space and making hard the understanding of FBs' interaction.

However, IEC 61499 provides a solution based on the *Adapter* concept [2, pp. 39-43]. This concept allows grouping together events and data to form an interface that is represented as a FB. It is similar to the concept of *Ports* used in UML 2.0. Ports simplify the interface of components by grouping of interface elements that logically belong together.

There are two different ways of using an adapter in a FB interface definition: as an interface accepting adapter called *Plug*, or as an interface providing adapter called *Socket*. In the interface definition of a FB the plugs are associated with the input side and sockets are associated with the output side. The definition of an adapter type is done from the plug perspective, because the plug side of an adapter connection is typically the requirements defining side. The socket has always the mirrored interface of the plug. That means events and data going into the plug are coming out of the socket and vice versa for the plug's outputs. Apart from reducing the number of connections the adapter concept has the great advantage that it increases the decoupling of application parts. The plug or socket user need no knowledge on the other part that they will be connected to.

We illustrate the use of the adapter concept in our example from Figure 2. Let us assume that our control loop is a pressure control loop. In this case our control algorithm would need the process value delivered from a pressure sensor. In order to make the control algorithm independent from a particular pressure sensor type, we define a pressure sensor adapter. Our control algorithm will initialize the sensor giving it a pressure range it should deliver. Furthermore we will request it for a new pressure value. The sensor should deliver the pressure in mPa and signal if there is any problem. The resulting adapter (from the plug view) can be seen in Figure 4(top). The usage of this adapter in the `Pressure_Control` FB and in the `Pressure_Sensor_TypeA` FB is shown in the lower part of this Figure. As the adapter has been defined according to the needs

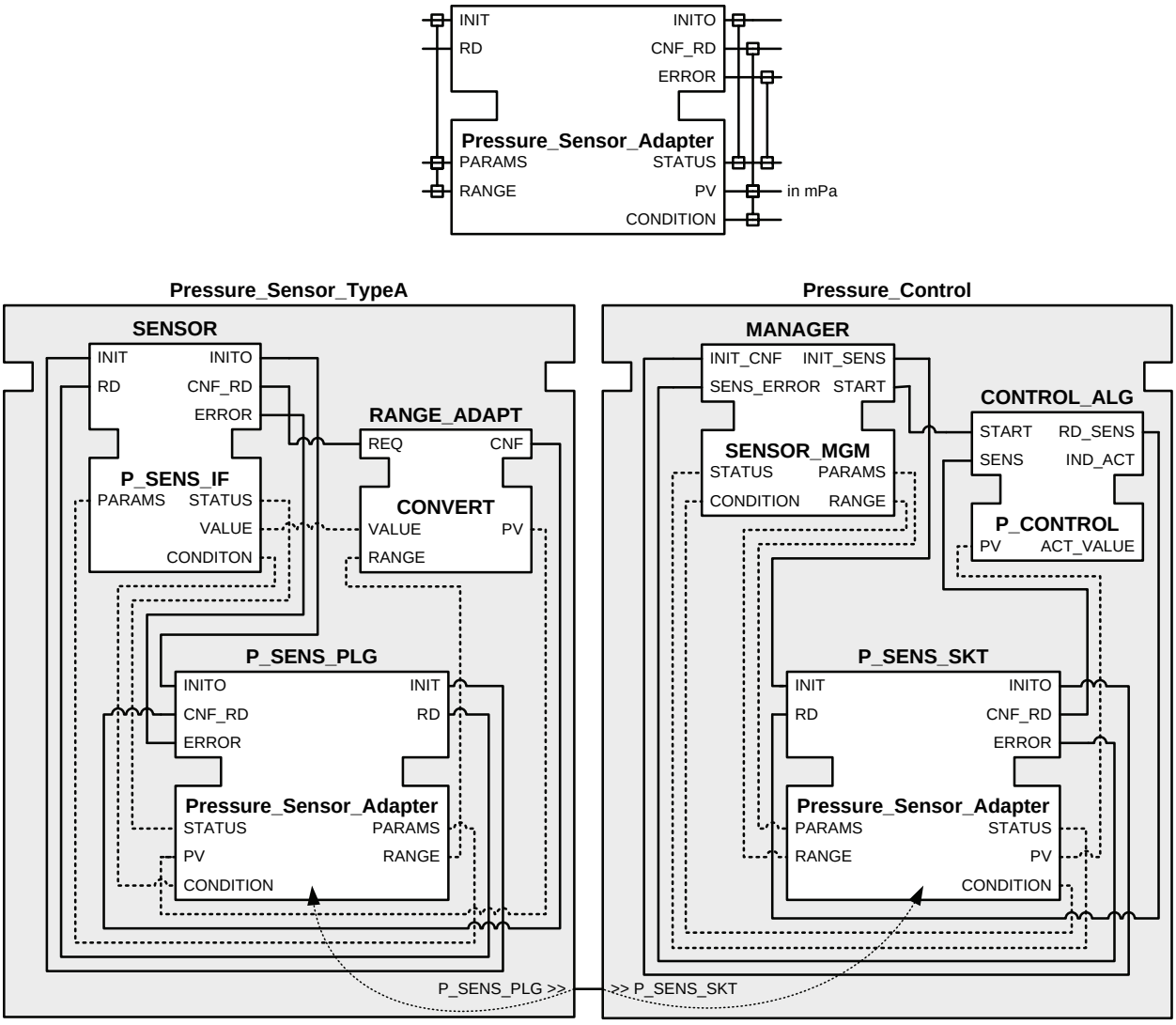

Fig. 4. An adapter Function Block for a pressure sensor and its usage in a pressure control application

of the pressure control algorithm, the plug can be directly used in the `Pressure_Control` FB. In `Pressure_Sensor_TypeA` there are some adjustments necessary. In our example the pressure sensor delivers its values not within the correct range. Therefore, the FB `RANGE_ADAPT` transforms the sensor value to the correct range. Note the mirrored interface of the socket in the `Pressure_Sensor_TypeA` FB.

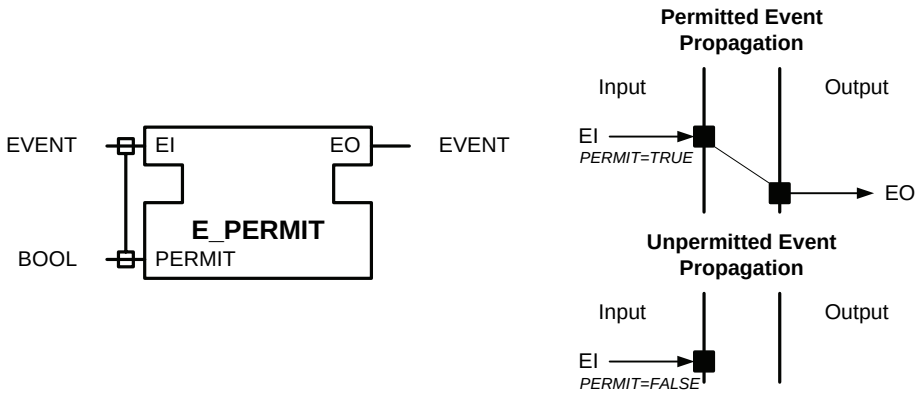

Fig. 5. Interface of the standard FB `E_PERMIT` as defined in Annex A of IEC 61499-1 and the time-sequence diagram describing its behavior

C. Describing Interface Semantics

Interface of a FB as defined in IEC 61499 is an important feature for designing safe and reusable software components. However, according to [24, pp. 50ff] the interface specification alone is not sufficient to ensure the use of a software component as a “black box” in different applications. Also the interface semantics have to be defined. The interface semantics describe how the interface has to be used (e.g., first event A, then event B), or how it behaves. For specifying the behavior of adapters and SIFBs IEC 61499-1 prescribes in Clause 6.1.3 the use of ISO/IEC 10731 time-sequence diagrams. According to this description the sequence of in- and output events and the values of data in- and outputs can be defined with time-sequence diagrams. Although the usage of time-sequence diagrams is prescribed for adapters and SIFBs, IEC 61499 does allow their use for the interface definition of any FB type (according to Annex A of IEC 61499-2). Figure 5 shows an example of such an interface behavior description.

This method provides additional means to document the FB behaviour to the benefit of both developers and users. Thus, the time-sequence diagrams can be used in the specification phase to define interfaces of new FBs. These specification can be passed then to the FB developer, who implements the functionality of the FB.

V. Compliance Profiles

One of the main goals of IEC 61499 development was to promote the development of heterogeneous systems composed of control devices of different vendors. Naturally such devices

may have completely different internals, but if they all are compliant to the standard, we can hope that some compatibility can be achieved. Unfortunately, the standard cannot foresee upfront all the features of devices' programming, configuration or communication that needs to be standardized. Instead, it defines a flexible and extensible mechanism of compliance profiles.

A compliance profile shall describe how the platform and implementation specific issues are solved. The structure of a compliance profile is described in IEC 61499-4. In general an IEC 61499 compliance profile has to define the following three points:

- The *Portability Provisions* describe how the models of IEC 61499 can be exchanged between tools of different vendors.
- The *Interoperability Provisions* describe how devices from different vendors can communicate with each other.
- The *Configurability Provisions* describe how devices from different vendors can be configured and how applications can be downloaded into the devices.

An example of a compliance profile is provided in [25].

The compliance profile concept has the great advantage that the standard can be extended to different needs and also define things the standard has intentionally left open (e.g., the concrete communication between devices on a certain field bus system). However this flexibility may also bring problems, as it opens the door to vendor specific extensions that destroy the open distributed system idea of IEC 61499. An example for this is the implementation of ISaGRAF, whose features follow the compliance profile mentioned in [26]. However, the compliance profile is not publicly available, so no other vendor can develop compatible solutions.

In our opinion, a regulatory instrument is needed to keep control system vendors in line with the ideas of IEC 61499. The international organization O^{3}neida has been volunteering so far to be such a regulatory body. O^{3}neida has been promoting the ideas of IEC 61499 and, on the other hand, has been involved in the development of IEC 61499 compliance profiles [27].

A good example of an issue targeted by a compliance profile is execution behavior (semantics) of FBs and applications built thereof. The reason for this are weak semantic-related descriptions in IEC 61499-1, which have been interpreted differently by different execution environment developers. This resulted in the situation that the same application can behave differently on different execution environments. Different execution problems were reported in [7], [8], [28]–[31]. To overcome these limitations, O^{3}neida is currently developing a compliance profile defining

the execution behavior of runtime environments and also clarifying the ambiguously defined elements of IEC 61499-1 (see [27] for more information). The compliance profile envisages three distinct execution models: sequential, parallel and cyclic. Its current state is overviewed in [32].

Many of the semantic ambiguities potentially can be fixed by amending the standard's text. Currently IEC 61499-1 is in a revision phase. The corresponding IEC working group (where the authors are members of) is actively working on improving several descriptions in the standard in order to provide a common and clearly defined execution behavior of applications and FB independent from the underlying execution behavior. Obviously, achieving consensus in the standard would be the best solution to satisfy both, the application developers and the device vendors.

VI. Conclusion

Currently control engineers are challenged by the growing complexity of automation projects, accompanied by shortened development time and tight quality requirements. New programming methodologies are necessary to increase software quality and reuse. With IEC 61499 modern software engineering methodologies have been adapted to the domain of industrial automation. This investigation showed that IEC 61499 defines several means which can help to improve software quality and reduce the development effort of distributed control systems. However, it also identified some open issues to be solved. How fast these issues can be solved will determine the success of IEC 61499.

Although the current adoption of IEC 61499 is low, it does not determine the failure of IEC 61499. One reason for the low adoption rate so far has been that control engineers needed comprehensive solutions rather than single technologies. Such solutions have finally appeared. The solutions of ICS Triplex, nxtControl, and the open source project 4DIAC are the promising signs towards broad industrial application of IEC 61499.

References

[1] Iacocca Institute, "21. Century Manufacturing Enterprise Strategy: An Industry-Led View," Iacocca Institute, Bethlehem, Pennsylvania, Tech. Rep., 1991.

[2] IEC 61499-1, *Function blocks – Part 1: Architecture*. Geneva: International Electrotechnical Commission, 2005.

[3] IEC 61499-2, *Function blocks – Part 2: Software tool requirements*. Geneva: International Electrotechnical Commission, 2004.

[4] IEC 61499-4, *Function Blocks – Part 4: Rules for compliance profiles*. Geneva: International Electrotechnical Commission, 2005.

[5] A. Zoitl, T. Strasser, K. Hall, R. Staron, C. Sünder, and B. Favre-Bulle, "The past, present, and future of IEC 61499," in *3^{rd} Intl. Conf. on Industrial Applications of holonic and Multi-Agent-Systems, HoloMas*, Regensburg, Germany, 2007.

[6] A. Zoitl, T. Strasser, C. Sünder, and T. Baier, "Best of both worlds: IEC 61499 in harmony with IEC 61131-3?" *to apear in: Industrial Electronics Magazine*, 2009.

[7] C. Sunder, A. Zoitl, J. Christensen, V. Vyatkin, R. Brennan, A. Valentini, L. Ferrarini, T. Strasser, J. Martinez-Lastra, and F. Auinger, "Usability and interoperability of IEC 61499 based distributed automation systems," in *Industrial Informatics, 2006 IEEE International Conference on*, Aug. 2006, pp. 31–37.

[8] V. Vyatkin and V. Dubinin, "Sequential axiomatic model for execution of basic function blocks in IEC61499," in *5th IEEE International Conference on Industrial Informatics (INDIN'07)*, Jul. 2007.

[9] R. Lewis, *Modeling Control Systems Using IEC 61499 - Applying Function Blocks to Distributed Systems*. London: The Institution of Electrical Engineers, 2001.

[10] V. Vyatkin, *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design*. USA: ISA-o3neida, 2007.

[11] IEC 61131-3, "Programmable controllers – Part 3: Programming languages," International Electrotechnical Commission, Geneva, Tech. Rep., 1993.

[12] A. Zoitl, *Real-Time Execution for IEC 61499*. USA: ISA and O^{3}neida, 2009.

[13] ICS Triplex ISaGRAF Inc., "ISaGRAF User's Guide," Online Available: `http://www.isagraf.com`, Jun. 2009.

[14] V. Vyatkin and J. Chouinard, "On comparisons of the isagraf implementation of IEC 61499 with FBDK and other implementations," in *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on*, Jul. 2008, pp. 289–294.

[15] D. Lavallée, J.-F. Laliberté, N. Landreaud, K. Thramboulidis, P. Bettez-Poirier, F. Desy, F. Darveau, N. Gendron, and C.-D. Trang, "An IEC 61499 configuration with 70 controllers; challenges, benefits and a discussion on technical decisions," Online Available: `http://www.isagraf.com/pages/documentation/ETFA07_SS1_Final17Oct2007.pdf`, Jun. 2009.

[16] nxtControl GmbH, "nxtControl - next generation software for next generation customers," Online Available: `http://www.nxtcontrol.com/`, Jun. 2009.

[17] Object Management Group, "Model Driven Architecture," Online Available: `http://www.omg.org/mda/faq_mda.htm`, Jun. 2009.

[18] B. Huber, R. Obermaisser, and P. Peti, "MDA-based development in the DECOS integrated architecture - modeling the hardware platform," in *Object and Component-Oriented Real-Time Distributed Computing, 2006. ISORC 2006. Ninth IEEE International Symposium on*, April 2006, pp. 10 pp.–.

[19] V. Vyatkin, H.-M. Hanisch, C. Pang, and J. Yang, "Application of closed-loop modelling in integrated component design and validation of manufacturing automation," *IEEE Transactions on Systems, Machine and Cybernetics - C*, vol. 39, pp. 17–28, 2008.

[20] T. Strasser, C. Sünder, M. Rooker, O. Hummer-Koppendorfer, A. Zoitl, and I. Müller, "Enhanced IEC 61499 system model

for evolution of control applications in distributed industrial-process measurement and control systems," in *European Control Conference*, Jul. 2007.

[21] 4DIAC Consortium, "Framework for Distributed Automation and Control (4DIAC)," Online Available: http://www.fordiac.org, Jun. 2009.

[22] T. Strasser, M. Rooker, G. Ebenhofer, A. Zoitl, C. Sünder, A. Valentini, and A. Martel, "Structuring of large scale distributed control programs with IEC 61499 subapplications and a hierarchical plant structure model," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2008*, Sept. 2008.

[23] C. Sünder, A. Zoitl, J. Christensen, M. Colla, and T. Strasser, "Execution models for the IEC 61499 elements composite function block and subapplication," in *Industrial Informatics, 2007 5th IEEE International Conference on*, vol. 2, Jun. 2007, pp. 1169–1175.

[24] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, 2nd ed. New York: ACM Press, 2002.

[25] J. H. Christensen, "IEC 61499 Compliance Profile for Feasibility Demonstrations," Online Available: http://www.holobloc.com/doc/ita/index.htm, Jun. 2009.

[26] TÜVRheinland, "ISaGRAF 5.1 assessment according to IEC 61499," Online Available: http://www.isagraf.com/pages/documentation/TUV/ISaGRAF_5-1_1499-TUVreport.pdf, Jun. 2009.

[27] O^{3}neida, "Compliance Profile," Online Available: http://www.oooneida.org/standards_development_Compliance_Profile.html, Jun. 2009.

[28] L. Ferrarini and C. Veber, "Implemenation approaches for the execution model of IEC 61499 applications." in *Proceedings of the 2nd IEEE International Conference on Industrial Informatics*, Berlin, Jun. 2004, pp. 612–617.

[29] V. Vyatkin, V. Dubinin, L. Ferrarini, and C. Veber, "Alternatives for execution semantics of IEC61499," in *5th IEEE Conference on Industrial Informatics (INDIN'07)*, 2007, pp. p.1151–1156.

[30] K. Thramboulidis and G. Doukas, "IEC 61499 execution model semantics," in *Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecommunications*, 2007, pp. 223–228.

[31] *Special Session: S07 Execution Semantics of IEC 61499 Function Block Applications*, vol. 2, Jun. 2007.

[32] V. Vyatkin, "The IEC 61499 standard and its semantics," *to apear in: Industrial Electronics Magazine*, 2009.