Automation Components for Complex IEC 61499 Compliant Systems

Federico Pérez, Isidro Calvo, Fabian López
Dept. of Automatic Control and Systems Engineering
University of the Basque Country (UPV/EHU)
Bilbao, Spain
{federico.perez, isidro.calvo, fabian.lopez}@ehu.es

Ismael Etxeberria-Agiriano
Dept. of Computer Languages and Systems
University of the Basque Country (UPV/EHU)
Vitoria-Gasteiz, Spain
ismael.etxeberria@ehu.es

***Abstract*—In the past, automation systems were hardly considered for changes during their lifetime due to their complexity. However, nowadays, their integration with other systems in time is considered as an important issue at design. In addition, automation applications demand new capabilities such as dynamic configuration or run-time reconfiguration in response to process changes or alarms. New methodologies are necessary in order to face these requirements. Component-based technologies, originally introduced in the software engineering industry aimed at obtaining efficient, structured and reusable designs, proved to be useful. In this paper we introduce an IEC 61499-based framework capable of dealing with reconfiguration issues. A new abstraction layer is introduced to carry out communication between components: the *communication channel*. The proposed framework allows the combination of *automation components* and *communication channels* in order to ease the design and implementation of complex IEC 61499 compliant distributed systems.**

***Keywords*—IEC 61499, Automation components, communication channels, reconfiguration**

I. Introduction

Nowadays, companies must be able to adapt quickly their productive processes to the changes demanded by the market. This fact requires that modern industrial engineering methodologies should be capable of introducing new characteristics such as adaptability, reusability and reconfiguration at run-time.

In this scenario, the use of advanced development tools has become mandatory for designing the new automation systems. Actually, providing better engineering tools has been a hot research topic during the last years. These new tools should provide mechanisms that ease the design and development of new automation applications from different perspectives such as: (1) providing higher integration, (2) automating sequential tasks during the engineering process, (3) easing the compatibility among different development environments and (4) increasing the reusability of the applications.

Since, modern control applications have gained in complexity, traditional design methodologies based on modular programming have proven not adequate. On the contrary, new approaches require the abstraction of complex structures by means of simple elements.

Besides, for legacy reasons new development tools must keep backwards compatibility with established standards. Currently, two major standards are used in the automation field: one for the programming of control devices and another for the design of distributed control systems. Those standards are IEC 61131-3 [1] for PLC programming, and IEC 61499 [2], [3], aimed at the design of distributed control systems. Unfortunately, in both cases more or less traditional programming techniques are still in use. In these methodologies program development is typically implemented tightly to the underlying infrastructure. This way, resulting applications tend to be fully dependent on the final hardware used complicating the reusability of the applications.

Recently, some innovations aimed at making automation projects compatible among different platforms have been presented for IEC 61131-3. Some of these works use methodologies based on XML models [4]. This approach allows defining compatible PLC projects for different suppliers.

According to the IEC 61499 standard, FBs (Function Block) are the main building blocks for creating control systems. Thus, applications are created by connecting FB blocks that wrap several devices/resources linked by different network communications [5], [6].

In the software engineering domain, components are self-contained entities or subsystems that can be combined together like building blocks to design more complex systems. This approach ensures that the component declaration is independent is independent from its development [7]. Since reusability is one of the most relevant issues in component development methodologies, software components are intrinsically designed to be reused in different applications, separating the interface from its development.

In modern engineering methodologies, automation components are typically the starting point to achieve a highest abstraction level in the design of distributed control systems. Automation component-based networks introduce a higher level of abstraction to the distributed automation systems design. Also, it provides an abstract and hierarchical view of the plant that represents the typically hierarchic structure of the elements found in the production environments.

The rest of this paper is organized as follows. Section II presents some works related to the application of component-

based architectures in the automation arena. Section III introduces briefly the IEC 61499 standard. It also describes the main problem that this article tries to solve, i.e. the composition of the applications. Section IV presents the automation component concept with emphasis on communication channels. Section V describes IEC 61499 component-based automation pointing to the development methodology. Finally, Section VI concludes summarizing some remarks.

II. Related work

This section summarizes related work on structuring automation component architecture for industrial control applications. An automation component is a production software package or an automation module that encapsulates a set of related functions and data.

A. Traditional Component Architectures

The component-based architectures come and compete with object-oriented architectures. While object-oriented architectures are modeled based on real objects or imaginary objects they represent, in the component-based architecture components are generated based on prefabricated objects [8].

One of the main essential elements that differentiate components against objects is the interface concept. The interfaces are the media to access the components. A component must publish its interface to access it. An interface is a set of related callable functions. Components can specify separate interfaces and their implementation.

Most based on components works are related to Component-Based Software Engineering (CBSE). Against object orientation, CBSE concept is associated to real software problems. To this day, CBSE is more related to computer systems than with manufacturing systems. Works about CBSE are directly related to technologies based on Java RMI, CORBA-based standards and Microsoft COM-based standards [9], or more recently, with middleware based on SOA, ICE or DDS.

B. Automation Component Architectures

For some time there have been attempts to integrate component architectures in industrial automation systems.

Within this environment, three alternative types to the use of component based architectures are presented:

1. CBSE adaptation: There are several works related to this topic. These works develop automation architectures started from classic CBSE technologies. In this aspect some works that involve different technologies can be found in the literature, more specifically, related to RT-CORBA [10], DDS [11], [24] and SOA [12].
2. IEC 61131-3 programming: In this case traditional PLC architectures are used both for applications and to control engineering. Within this field both industrial solutions and academic solutions are presented. Between industrial solutions could be named Profinet-CBA [13] that uses Microsoft COM/DCOM technology. Moreover, as an example of academic solutions, Estevez in [14] uses XML as core in an integrated framework for a model-based design.
3. IEC 61499 programming: Within this standard, several component orientation technologies and engineering tools has been developed. Vyatkin at [15] introduces the term "Intelligent Mechatronic Component" (IMC) for the automation software design. This concept is very similar to "Technological Module" proposed in Profinet-CBA. An IMC is a machine or mechatronic component that is provided with preprogrammed software. Each IMC contains three elements: a physical mechatronic, a control device and a logic software component. Thramboulidis at [16] presents an Engineering Support System (CORFU-ESS) available to develop distributed control architectures. Within the system CASE tools are integrated that can be used to the development of FBs, mechatronic components, parameters, etc., providing a formal base using UML modeling. Cengic at [17] introduce the term "automation component" over IEC 61499 inspired in the concept of VHDL component. Black at [18] implements IMCs with IEC 61499 FBs within an approach to multi-agent systems with holonic principles that introduce an autonomous and collaborative behavior. Moreover, IEC 61499 tools like FBDK [19] o after that 4DIAC-IDE [20] allows the implementation of IEC 61499 system generating components through FBs. Moran at [21] presents a methodology to compose developments using this kind of tools emphasized the communication SIFBs to link the components. In this last work, a component is each one of standard elements such a FB, SIFB, resources, devices and application. Each FB can be thought of as a special type of component that encapsulates data and algorithms, as well as a state machine to control its execution. An application consists of a network of FBs and forms the whole system together with the communication network and the devices. Hametner [22] presents a scheme for the development of engineering applications and maintain in the control system and automation domain; In this work a new component architecture is specified that supports component oriented design, reusability and functional elements encapsulation.

III. IEC 61499 architecture

The IEC 61499 standard proposes an open architecture to design distributed automation applications. This standard has been defined to supports modular, reconfigurable and flexible distributed control systems.

The IEC 61499 standard specifies an architecture model for distributed applications in a generic way for systems automation.

The building block in IEC 61499 is the FB (Function Block). IEC 61499 extends the concept of FB in IEC 61131-3 with additional mechanisms of event management and distribution concept.

FBs are reusable software components that can be arranged in a network to build an application. Each FB has predefined its own set of inputs and outputs. The FB functionality is provided by means of algorithms that process input data and generating output data, while the event connections define the execution

order for the algorithms. The relations between events and algorithms execution is carry out by a transition state diagram called Execution Control Chart (ECC).

One application is composed by a FB network. Such application can be distributed over several devices linked by a communication network using Service Interface Function Blocks (SIFBs). One SIFB is a special type of FB that can be used to event and data transfer between the devices that compose the hardware structure over the network communications that links them.

The event and data connections in IEC 61499 applications are implemented easily in simple IEC 61499 applications. The distributed control applications development gets complicated for complex applications where the number of FBs and connection links multiplies.

The use of connecting lines to represent connections between FBs events and data through graphical elements is a good idea that is used in many of the graphic programming tools. However, the great number of these connecting lines prevents a clear view of the application and the system. From the point of view of the application developer, it is necessary to apply abstraction mechanisms that allow the application distribution both system point of view and network.

IEC 61499 uses the composite FB as mechanism to group FB networks so it is possible to distinguish functional elements within complex distributed applications. However, even using composite FBs, the large number of events and data connections between composite FBs makes the application get complex to distribute, modify or reconfigure.

At this point, it is necessary an abstraction higher level to identify the different elements/components of the application. This new abstraction level can be achieved through the use of automation components.

The design of applications using automation components network will allow a new installation view. The design of applications using automation components network will allow a new installation view. This will make possible to consolidate concepts such as maintainability, reusability, flexibility and reconfigurability.

IV. Automation Components

The automation components for IEC 61499 presented in this job are implemented through FB networks. These FBs can be both basic and composite. The main difference between FBs and automation components is that events and data intrinsically related are grouped in connection ports (Fig. 1).

It is possible to distinguish two types of ports: input ports (group set of event and data inputs) and output ports (group sets of event and data outputs). Events and data grouped in ports correspond to IEC 61499 events and data generally associated with the WITH modifier. This does not mean that any basic or composite FB is directly transportable to an automation component structure, but any automation component can be implemented by FBs or FBs IEC 61499 networks.

Communication port concept is comparable to the adapters interfaces FB in IEC 61499. However, unlike the adapters, these do not behave as interfaces in a mirrored form as in function for complex output (plugs) or complex inputs (sockets). In this case, the communication ports may have different sets of events and data in their twins interfaces.

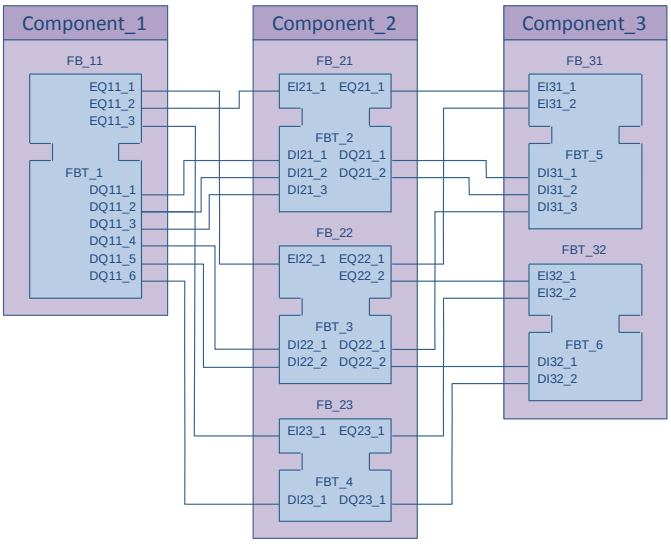

Fig. 1. FB-based programming to automation components.

Communication channels link the ports of different automation components that compose the application (Fig. 2). A communication channel is defined as a link mechanism for automation components that allows a higher abstraction in the IEC 61499 application development.

The distribution of the components over different devices does not prevent the mechanism of communication channels. Instead, this mechanism, available in both local applications and distributed applications, allows the abstraction of the mechanisms and communication networks.

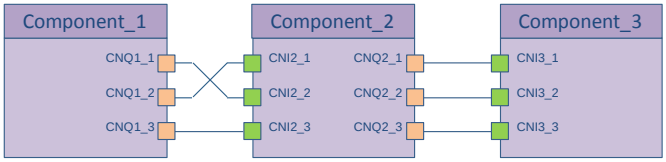

Fig. 2. Component based application.

The next revision of IEC 61499 standard places greater emphasis on communication means between technological devices separating the definition of the declaration system of communication segments. In this regard, the communication channels allow a step forward in the implementation of distributed control applications since they release to the application developer of the communication mechanisms between devices.

A. Communication Channels

Communication channels abstract the event and data communications in an application component-based.

From the point of view of the communication channel architecture in the communication channel are grouped events and data related (Fig. 3).

Communication channels in distributed control applications are intrinsically linked to the declaration, definition and configuration of the so called IEC 61499 communication segments.

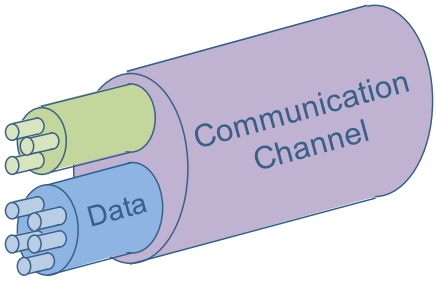

Fig. 3. Communication channel architecture.

Communication channels allow more services than event and data connections available in IEC 61499. Besides the transfer services for events and data values, the communication channels can be used for other functions:

- Synchronization for channel events and data
- Handling events and data
- Management for channel events and data
- Management of communication services
- Implementation of QoS policies
- Diagnosis of communication

From the point of view for application deployment, it is possible to consider two types of communication channels: local channels and network channels. Local channels are the ones that there are at the same device while networks channels employ communication networks for signal transmission.

Local communication channels can use different communication mechanisms depending of the distribution or not of the linked components between different resources in the same device.

In case of local communication channels that link components placed in the same resource, the implementation mechanism to the channels could be very easy because, in the simpler case, they corresponds with basic connections between output events and input events, and the basic connections between output data an input data of traditional IEC 61499 applications.

In case of local communication channels that link automation components placed in different resources, the implementation mechanisms for these channels are very similar to the network communication channels. They depend of communication mechanisms between resources that, in this aspect, can provide different types of devices.

Network communication channels join output ports with input ports of automation components that are placed on different devices. Obviously, devices that implement the distributed application are linked through a network of communication between them.

When the application is distributed across different devices or resources, communication channels allow to abstract communication mechanisms between such devices or resources.

The implementation of local communication channels between resources or network communication channels is delivered via SIFBs. This requires the existence of different SIFBs libraries each dependent mechanisms for the application deployment.

Network communication channels and local communication channels between resources translate into the final IEC 61499 system in SIFBs that implement communication mechanisms client/server based or publisher/subscriber based depending of the network that join device or the resources offered by devices at local level.

V. IEC 61499 Automation Component-Based

This section presents the automation systems using IEC 61499 and its development towards a component-based automation IEC 61499 compliant. It starts from the traditional IEC 61499 automation and the subsequent introduction of engineering tools to support the IEC 61499 programming.

In all cases the final product is a system IEC 61499 ready for download in a distributed control system. To do this, besides containing FBs networks that implement the control algorithm, the system must incorporate necessary SIFBs to implement communications between joined FBs, so that may be considered, remote connections.

A. Classic IEC 61499 Automation

In traditional IEC 61499 application development the resources for communication between distributed devices are provided by SIFBs. These SIFBs can be introduced on the application itself or after the mapping of the FBs making up the application on the resources included in the devices (Fig. 4).

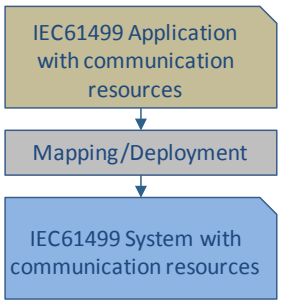

Fig. 4. Classic IEC 61499 system development.

The introduction of communication SIFBs manually on applications or once mapped on resources, poses an additional task to automation engineer. This task makes difficult the work of the engineer of automation, while introducing the control program must master the techniques of data exchange.

This type of approach makes difficult application maintenance, reuse or reconfiguring.

B. IEC 61499 Automation by Composer

The classic programming used in traditional IEC 61499 programming environments is very close to traditional IEC 61131-3 programming to PLCs. In a modern programming, engineering tools are used to facilitate the use of resources beyond the control algorithms.

Moran at [23] presents the Composer CASE tool to support the generation of systems with distributed control applications in IEC 61499 environments. The composer allows the introduction of communications SIFBs starting from a system with a functional application once deployed (Fig. 5). In this case the tool is able to recognize the functional distribution of the application and, at the same time, to introduce the necessary communication resources from a library of communications SIFBs. The result is an IEC 61499 system that melts functional applications and communication mechanisms.

The composer is a mechanism that, starting from IEC 61499 applications and application distribution between different devices and resources that make up the system, enables the development of new distributed applications with the communication mechanisms required for the application deployment.

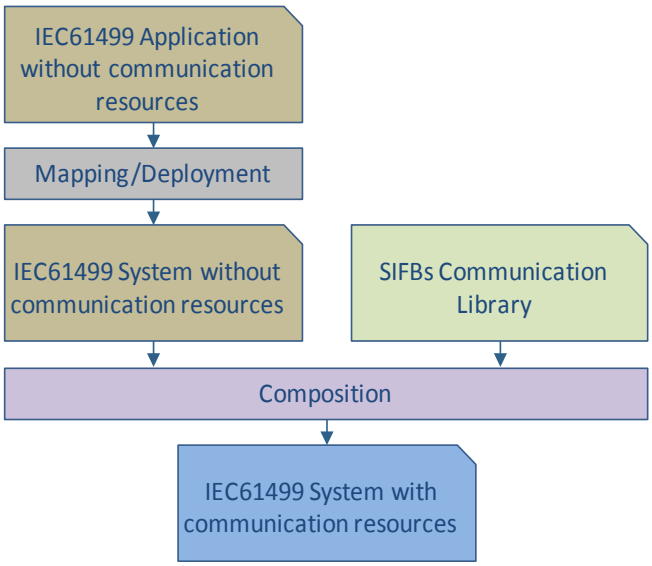

Fig. 5. IEC 61499 system development with composition.

From this point of view, the composer becomes a basic application that allows configuration of distributed control applications easily.

The composer can be considered as a tool prior to a control system as dynamic reconfiguration that allows reconfiguration of the distributed application independently of the own distribution of the functional control elements.

C. IEC 61499 Automation Component-Based

The component-based automation starts from an application which is expressed simpler than an IEC 61499 application. However, as in the IEC 61499 applications, it is necessary to map automation components on the devices/resources that make up the automation system. In this way it is obtained an automation system component based (Fig. 6).

The corresponding view to manipulate the system is not, as in the case of traditional IEC 61499 programming, the view of the resources, where mapped elements are observed in addition to the elements of the device/resource. In this case the essential system view corresponds with application view. No matter how functional elements (automation components) are distributed. The essential object is the application itself.

However, the component system includes the distribution and it is an essential element from the operational standpoint in the distributed control system.

In order to obtain an IEC 61499 system without observed communication resources, it is necessary a translation tool. This tool starts from component based system and from a library of automation components. This library contains the definition of automation components that allows the implementation of its instances.

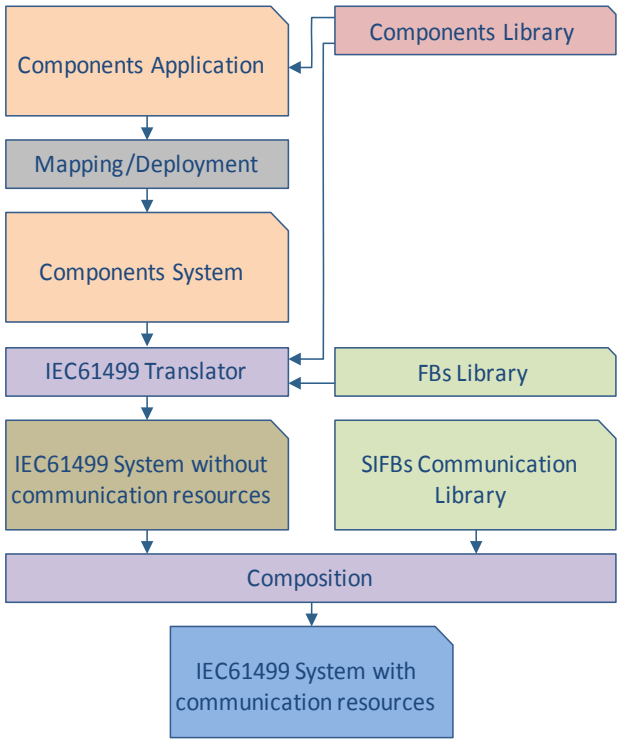

Fig. 6. IEC 61499 system development with components, translate and composition.

Once obtained the IEC 61499 system without communication resources, it is possible to use the same tool composer that used in the previous section (Fig. 5). Thus, it is possible to obtain a complete IEC 61499 system which contains the needed communication resources.

As seen, in case of distributed applications using automation components, it is possible to use the composer, starting from the generated system with the translation tool. However, this procedure is not optimal from the point of view of the generation of the final system. Composer tool depends largely on SIFBs library and lose many of the advantages of

working with communications channels. Additionally, the composer tool is too general and introduces too much overhead on the connections between devices.

Therefore, it is more optimal to use a new composer that works directly with the distributed components instead of dealing with distributed FB networks, as shown in Fig. 7.

The proposed development environment (framework) to the generation of the IEC 61499 system with communication resources, shown in Figure 7, starts from three interface editing tools:

1. Channel Editor: The channel editor allows generating XML interfaces for communication channels. Jointly with the communication channels, the interfaces to the communication ports of the automation components are created. Input ports may differ from the output ports for the same channel in terms of the services they can add to their own channels (data and event handling, quality of service, etc.). The result of editing the channel is a XML interface that integrates within a library of communication channels.
2. Component Editor: The component editor generates the XML interfaces for automation components. The component editor takes information from the library of channels and the interfaces in the IEC 61499 FB library. The FBs library is the library comes from IEC 61499 compatible editors. The result of the edition of a component is a XML interface that integrates within a library of automation components.
3. Communications Editor: It generates the XML interfaces for communication mechanisms between components. The result of communication editing is an XML interface that integrates FB and SIFB networks for communication. This library includes both local and remote communication mechanisms. The component editor takes information from the channels library and the compliant IEC 61499 SIFB library. Furthermore, in the same interfaces information corresponding to the generation of the compliant IEC 61499 segments communication interfaces is also introduced.

The component application editor allows composing an application from the components library. The edition of the application is independent from its deployment. Only use the component library comes from the component editor. Ports between components are connected by communication channels regardless of the final distribution of the components. The result is applications much easier to handle and easily reconfigurable.

The deployment tool allows editing the system. It is able to instantiation of devices, resources in devices and mechanisms and communication networks. It starts from the IEC 61499 compliant devices and resources library and the communications library created with the communication editor.

The deployment tool also allows mapping the automation components on the resources that belongs to the devices.

The result of applying the deployment tool on a component application is a system based on automation components.

The deployment tool provides two views from the automation component-based system: the project view and the network view. In the project view the devices with their resources and component instances deployed on each resource are shown. This allows quick access to the automation components for diagnosis and maintenance.

Moreover, in the network view can be observed the control devices distribution and their resources on different network segments. It is needed that resources will appear because it is possible to define different communication mechanisms between resources within the same device. In the same way than the project view, the network view allows quick access to devices and resources for diagnosis and maintenance.

Finally, the component composer generates a full IEC 61499 system with communication resources included. This system is IEC 61499 compliant and, therefore, can be used by any IEC 61499 compliant tool.

The component composer starts from the components system generated by the deployment tool, but should have access to the information contained in the compliant IEC

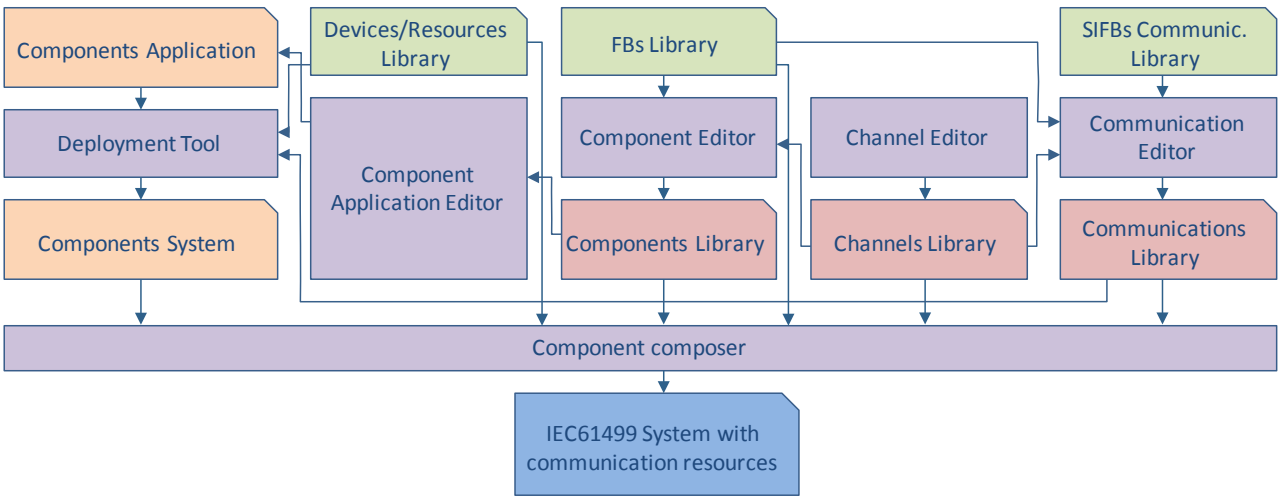

Fig. 7. IEC 61499 system development with components, translate and composition.

61499 libraries and the libraries used in the different developed processes during editing.

As indicated previously, the new generated compliant IEC 61499 system is much more optimized than that the generated by the composer explained in Section C. This is because the method of communication channels allows a much more efficient mechanism than the SIFBs direct association with networks communication used by the previous composer.

VI. Conclusions

A component based development methodology for distributed control systems has been presented in this paper. This methodology eases the generation of complex IEC 61499 compliant systems by means of two major entities: automation components and communication channels.

The use of automation components provides a higher abstraction level while, at the same time, increases reusability and adaptability possibilities.

The communication channel entity allows abstracting any type of communication mechanism during the application development stage, easing the future lifecycle of the applications, since different communication technologies could be exchanged in a transparent way. This fact provides significant advantages over traditional methods for developing distributed applications under the IEC 61499 standard. Furthermore, the presented approach eases, improves and encourages new application reconfiguration mechanisms, even at run-time.

Finally, in the opinion of the authors, the above-described development environment allows the creation of complex distributed control applications by using the here proposed entities. Thus, applications based on automation components can be edited by any IEC 61499 compliant editor such as FBDK or 4DIAC-IDE.

Acknowledgment

This work has been supported by the Spanish Ministry of Economy and Competitiveness (MINECO) under project DPI2012-37806-C02-01 and the University of the Basque Country (UPV/EHU) under EHU11/35 and EHU13/42 projects.

References

[1] International Electrotechnical Commission, “IEC 61131-3: Programmable Controllers. Part 3: Programming Languages”, International Standard, 2003.

[2] International Electrotechnical Commission “IEC 61499-1: Function Blocks - Part 1 Architecture”, International Standard, First Edition, Geneva, 2005.

[3] International Electrotechnical Commission “IEC 61499-2: Function Blocks - Part 2 Software tool requirements”, International Standard, First Edition, Geneva, 2004.

[4] Marcos, M., Estévez, E., Pérez, F., Van Der Wal, E., “XML Exchange of Control Programs”, IEEE Industrial Electronics Magazine, Vol.3, pp. 32-35, 2009.

[5] Lewis. R., Modelling control systems using IEC 61499, Control Engineering Series 59. The Institution of Electrical Engineers, 2001.

[6] Zoitl, A., Vyatkin, V., “IEC 61499 Architecture for Distributed Automation: The “Glass Half Full” View”, IEEE Industrial Electronics Magazine, Vol.3, No. 4, pp. 7-23, 2009.

[7] Szyperski, C., Component Software: Beyond Object-Oriented Programming, Addison-Wesley Professional, 2002.

[8] Heineman, G.T., Councill, W.T., Component based software engineering: putting the pieces together, Addison-Wesley Professional, 2001.

[9] Hull, M.E.C., “Approaches to component technologies for software reuse of legacy systems”, Computing & Control Engineering Journal, Vol. 12, Issue 6, pp. 281-287, 2001.

[10] OMG, Object Management Group, “Real Time-CORBA Specification”, Version, 1.2, January 2005. Available at: www.omg.org , 2004

[11] OMG, Object Management Group, “Data Distribution Service for real-time systems, version 1.2”, 2007. Available from www.omg.org, 2007.

[12] Tsai, W.T., Lee, Y.H., Cao, Z., Chen, Y., Xiao, B., “RTSOA: Real-Time Service-Oriented Architecture”, 2nd IEEE Int. Service-Oriented Syst. Eng. (SOSE’06), pp 49-56, 2006.

[13] Profinet-CBA, Profibus-Profinet International, WebSite: http://www.profibus.com/, last access April 2012.

[14] Estévez, E., Marcos, M., Orive, D., “Automatic generation of PLC automation projects from component-based models”, International Journal of Advanced Manufacturing Technology, Vol. 35, Nº. 5-6, pp. 527-540, 2007.

[15] Vyatkin, V., Intelligent mechatronic components: control system engineering using an open distributed architecture, Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA’03), vol. 2, pp. 277-284, 2003.

[16] Thramboulidis, K., “IEC 61499 in Factory Automation” Proceedings of the International Conference on Industrial Electronics, Technology and Automation, CISSE-IETA 05, 2005.

[17] Cengic, G., Ljungkrantz, O., Akesson, K., “A Framework for Component Based Distributed Control Software Development Using IEC 61499”, Proceedings of the 11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA’06), pp. 782-789, 2006.

[18] Black, G., Vyatkin, V., Intelligent Component-Based Automation of Baggage Handling Systems With IEC 61499, IEEE Transactions on Automation Science and Engineering, vol. 7, issue 2, pp. 337-351, 2010.

[19] Tutorials- Basic Concepts of IEC 61499 Tutorial FBDK (Function Block Development Kit) Website: http://www.holobloc.com/, last access: April 2011.

[20] 4DIAC - Open Source for Distributed Industrial Automation, WebSite: http://www.fordiac.org/, last access April 2011.

[21] Morán, G., Pérez, F., Estévez, E., Orive, D., Marcos, M., “Achieving Distributed Control Applications Using IEC 61499 and Communication Standards”, 43th CIRP Conference on Manufacturing Systems (CIRP'2010), pp. 1028-1035, 2010.

[22] Hametner, R., Zoitl, A., Semo, M., “Automation Component Architecture for the Efficient Development of Industrial Automation Systems”, 6th annual IEEE Conference on Automation Science and Engineering, pp. 156-161, 2010.

[23] Morán, G., Pérez, F., Orive, D., Estévez, E., Marcos, M., “Automatic Composition of IEC 61499 Distributed Control Applications”, 16th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA’11, pp. 1-7, 2011.

[24] Calvo, I., Pérez, F., Etxeberria-Agiriano, I., García de Albéniz, O. “Designing High Performance Factory Automation Applications on Top of DDS”, International Journal of Advanced Robotic Systems, vol. 10, no. 205, 2013.