Control Communications with DDS using IEC61499 Service Interface Function Blocks

Isidro Calvo[1], Federico Pérez[1], Ismael Etxeberria[2], Guadalupe Morán[3]

[1,2]{isidro.calvo, federico.perez, ismael.etxeberria}@ehu.es
[1]Dept. of Automatic Control and Systems Engineering
[2]Dept. of Computer Languages and Systems
University of the Basque Country (UPV/EHU)
SPAIN

[3]gmoran@tol.mx
Electronics Division
Faculty of Engineering
UAEM, Toluca
MEXICO

Abstract

The IEC61499 is an open standard for distributed control and automation. The interface between control software and hardware or communications is achieved by means of the so-called Service Interface Function Blocks (SIFB). This paper presents the guidelines to build communication SIFBs based on the emerging OMG DDS (Data Distribution Service) middleware. This specification implements in a very efficient way the Publisher/Subscriber paradigm providing significant QoS configuration possibilities. These characteristics make DDS suitable for implementing the communications among time-critical devices. By using these DDS-SIFBs within IEC61499 code generation tools, the designers of the distributed applications will be allowed to use this powerful technology in the new distributed applications.

1. Introduction

Traditionally, manufacturing systems were configured following centralized approaches where the process control resided in central controllers such as PLCs or Industrial PCs which used I/O communications to acquire or deliver real-time process data. Nowadays, however, manufacturing systems are demanding increasingly flexible, agile and adaptable solutions. These new manufacturing devices are typically distributed control systems that do not only require I/O communications but also horizontal (among different controllers involved in the process) and vertical (with other nodes of the factory) communications.

This scenario requires adopting new approaches. For example, new embedded hardware platforms are being used in factory automation. These new platforms allow the introduction of modern techniques, already used in the embedded applications domain, to improve the scalability, agility and reconfiguration of the new industrial systems.

Also, new standards are being adopted in the new industrial systems such as the IEC 61131 standard which still follows a centralized approach for building factory automation applications. The newer IEC 61499 is designed for distributed control systems where several applications can share several devices and resources within complex manufacturing system architectures.

With regards to industrial control communications, different middleware technologies such as Web Services, OPC (OLE for Process Control) [13] and CORBA [14] have been proposed in order to ease the integration of control devices which run over heterogeneous platforms. However, most of these middleware technologies, which are based on the Client/Server model, do not adapt properly to some characteristics of the real-time process data, namely periodic messages with data sampled values. This work analyzes the application of the emerging DDS (Data Distribution Service) [10] to the industrial control communication field as IEC 61499 SIFBs. DDS is a relatively new middleware specification based on a publisher/subscriber model which is gaining acceptation in several industrial domains. As a consequence, DDS may be seen as a kind of virtual bus which abstracts the underlying network, providing mechanisms to specify certain QoS parameters according to the characteristics of the underlying network infrastructure.

DDS middleware specification uses certain interesting characteristics such as platform independence (processor architecture, programming language and operating system) as well as a generic definition for data and operations through Interface Definition Language (IDL) definitions. This virtual bus based on middleware is an interesting alternative for the construction of scalable and modular distributed applications, providing a fair compromise between system performance and development and maintenance effort.

The layout of the paper is as follows. Section 2 presents a brief description of the IEC 61499 standard. Section 3 provides an overview of the Data Distribution Service (DDS). Section 4 indicates a general methodology to create SIFBs and the set of SIFBs for DDS communication system developed following it. The paper finishes with some concluding remarks.

978-1-4244-6850-8/10/$26.00 ©2010 IEEE

2. IEC 61499 Standard

The IEC 61499 standard [1, 5, 6] is an open component oriented architecture designed for the development of distributed control and measurement systems. This standard defines a generic distributed model architecture that provides a reference model for the use of Function Blocks (FBs) in distributed industrial applications.

Function blocks are the basic building block from which entire applications are built. They are functional units of software comprising an individual instance or copy within a resource [7]. They can be distributed and interconnected across multiple controllers. Basically, every function block has event inputs and outputs as well as data inputs and outputs. In a basic function block the execution of an algorithm is triggered by the occurrence of an input event. This algorithm typically produces new output data. When the algorithm is finished an output event is generated which may be the input event to another function.

There are specialized types of FBs (Service Interface Function Blocks - SIFBs) that are able to access hardware resources, communications resources, management, etc. The authors propose using these SIFBs for all types of communications, both I/O communications as well as horizontal process communications. This mechanism offers a standard interface for programming communications which is application independent. This approach allows combining this procedure with other mechanisms as proposed in refs [8, 9].

3. Data Distribution Service (DDS)

The Object Management Group (OMG) has released recently the Data Distribution Service (DDS) [10] as a platform-independent standard for data centric publish/subscriber middleware systems. The standard ensures deterministic data exchange and allows the management of many aspects of the communication behavior to meet application requirements. As opposed to message-oriented approaches, DDS does not exchange data in the form of messages, but on the contrary, data is formally defined in a platform independent way. This approach allows the automatic generation of communication source code for specific target platforms.

DDS may combine different programming languages (mainly C, C++ and Java) and operating systems (such as Windows, VxWorks, QNX, Lynx and other Unix/Linux derivatives).

This commercial off-the-shelf technology has been adopted in many military and civil sectors: aerospace, traffic control systems, SCADA, robotics and automation [15].

DDS defines a virtual Global Data Space that may be accessed by applications from heterogeneous platforms. With a pub/sub many-to-many paradigm, publishers produce/write information that can be consumed/read by subscribed applications which are decoupled in time, space and synchronization. This means that an application joining the system may subscribe and obtain persistent data that was issued at an earlier moment. This general communication idea is illustrated in Fig. 1.

Figure 1. DDS Global Data Space.

The data information unit that flows from publishers to subscribers within this model is the topic. Applications declare their intention to publish topics and to subscribe to topics on a domain, as depicted in Fig. 2.

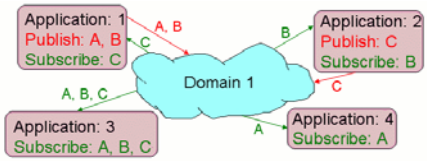

Figure 2. Topics in domain 1.

DDS defines the Data-Centric Publish-Subscribe (DCPS) model, a topic-based standard API involving the following entities:

- **Domain** – Defines a communication context which provides a virtual environment, isolating different concerns and thereby optimizing communications.
- **Topic** – Is the information unit that atomically connects publishers and subscribers, identified by a name, allowing anonymous and transparent communications. Topic instances are data objects identified by a key, defined using a subset of (IDL) [10].
- **Domain Participant** – Represents the entity taking part of an application in a DDS domain.
- **Data Writer** – It declares the intent to publish a topic and provides type-safe operations to write/send data.
- **Data Reader** – It declares the intent to subscribe a topic and provides type-safe operations to read/receive data.
- **Publisher** – It is created by a domain participant to manage a group of data writers.
- **Subscriber** – It is created by a domain participant to manage a group of data readers.

The middleware dynamically detects new participants in the system and establishes connections between publishers and subscribers for a specific topic. There are two different approaches for this:

1. **Callback** – This is a listener-based non-blocking asynchronous mechanism.
2. **Waitset** – This is a blocking synchronous mechanism that awakes an application when a desired condition is met.

There are several QoS properties that can be configured at different DDS entity levels: topics, data writers, data readers, publishers and subscribers. Some of the most relevant parameters are:

- **Durability**: defines if data is kept for late joiners and for how long.
- **Deadline**: specifying an expiration time
- **Update frequency**
- Maximum data delivery **latency**
- Data **delivery priority**
- Data **delivery reliability**
- **Content awareness**, allowing the use of filter expressions for data readers

4. SIFBs for DDS

The IEC 61499 standard defines a service as a functional capability of a resource which can be modeled by a sequence of service primitives (abstract representation of an interaction between an application and a resource). As a consequence, a SIFB is an abstraction of a software component that implements some services. In this sense, SIFBs are FBs that provide one or more services to an application but offering a standard interface to the application.

This section presents the development of a set of SIFBs for DDS using the 4DIAC [3] tool. Firstly, the main steps to build 61499 SIFBs with any development tool are outlined.

4.1. Steps to develop IEC 61499 SIFBs

Nowadays, there are few software tools compliant to the IEC 61499 standard. Some of them are commercial tools like ISaGRAF ICS Triplex and TCS (NZ) DI of TAIT [1]. Others are freeware, mostly used in research environments, like FBDK, FBench or 4DIAC. In particular, 4DIAC is an initiative to provide an open, IEC 61499 standard compliant environment, which is based on the targets portability, configurability and interoperability.

As commented before, the SIFBs aim at communicating data values and events between resources. In fact, SIFBs are necessary when any form of interaction between a FB within a resource and the external world are required. If necessary the SIFBs are used to invoke hardware related functions.

Fig. 3 shows the general steps for developing SIFBs to be used within an IEC 61499 standard compliant C++ tool. Basically, these are: (1) Identification of the service the SIFB should implement; (2) definition of the SIFB inputs and outputs, both event and data; (3) specification of the primitives of the service within the IEC61499 tool and (4) development of the algorithms that implement the service.

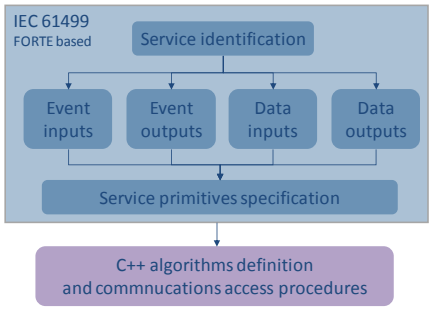

Figure 3. Elements in a SIFB development.

The whole methodology implies the use of several tools during the development: (1) An IEC61499 tool to define the SIFB services (here the 4DIAC-IDE) and (2) a C/C++ development platform (here, Eclipse with CDT, a free multiplatform development environment). Fig. 4 illustrates the general scenario for developing SIFBs using 4DIAC, Eclipse and a C/C++ tool.

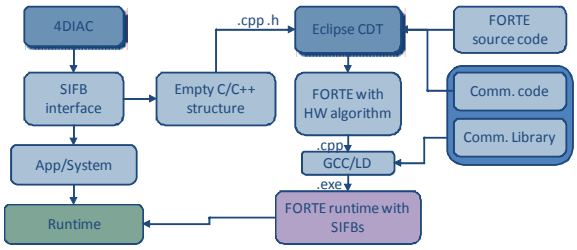

Figure 4. Steps for software development.

4DIAC is used to define a new SIFB, its input and output data and events, as well as the definition of the service primitives. The structure of the SIFB can be exported into a .cpp/.h file. The Eclipse CDT tool is used to declare and define the IEC61499 methods that link the algorithms to the hardware. The C/C++ IDE generates a project in which the access to the communications is programmed. This project includes the runtime source code to link with the developed communication code. In the last stage a new runtime associated with the SIFB that includes the hardware access library is created. The new runtime file implements and fulfils the services of the SIFB set.

4.2. SIFBs for DDS communication

The previous steps have been followed to implement SIFBs for DDS communication. In particular, OpenSplice DDS [16] has been used to design a set of SIFBs that allow publish/subscribe communications. Figure 5 shows two of the SIFBs implemented following the proposed methodology.

In order to use these blocks it is necessary to specify certain information regarding the topic name, its type, only basic types and certain parameters of quality of service are supported in the current implementation.

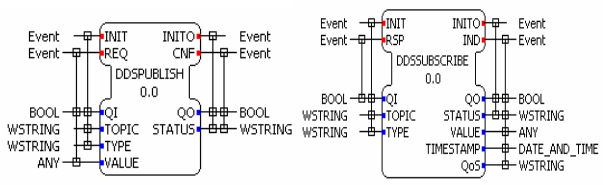

Figure 5. DDS publisher and Subscriber SIFBs.

Fig. 6 illustrates the design of a typical application from the design tool. In this application a topic called *TestEnable* is published by the Supervisor using the DDSPUBLISH SIFB. The controller is subscribed to this topic by using the DDSSUBSCRIBER SIFB. Similarly, a second topic called *CtrlValue* is published by the Controller and the Supervisor subscribes to its data.

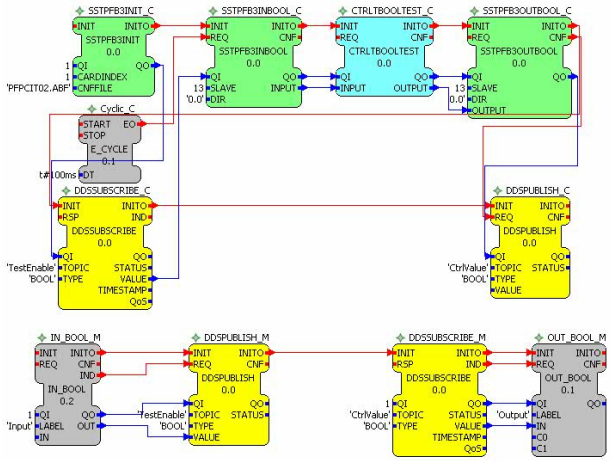

Figure 6. Application configuration.

5. Conclusions

SIFBs are the interface between applications and the hardware of embedded devices. In fact, they link the resources to the process and communication sub-systems.

This paper presents the steps to design SIFBs for data access technologies using the SIFB template provided by the IEC 61499 standard. This methodology provides several benefits to system engineers of the industrial control field, such as reducing development costs and reducing the duration of the system implementation.

Several middleware technologies have been analyzed in order to provide a virtual bus into which basic software components may be used, so the construction of new distributed applications is simplified. In this work the use of DDS is proposed to create IEC 61499 SIFBs components that may be integrated in the development environment. DDS provides very interesting characteristics for industrial environments such as efficient implementations, platform independence and a set of mechanisms to control QoS parameters. The authors are implementing a set of SIFBs that use the DDS middleware on top of GNU/Linux Debian with commercial hardware.

6. Acknowledgements

This work has been financed in part by the University of the Basque Country through grant EHU09/29, and MICINN under project DPI2009-08102.

References

[1] http://www.oooneida.org

[2] http://www.omg.org

[3] http://www.fordiac.org/

[4] http://www.holobloc.com/Tutorials, *Basic Concepts of IEC 61499 Tutorial FBDK*

[5] International Electro-technical Commission. International *Standard IEC 61499-1 Function Blocks International Electrotechnical Commission, Part 1*: Technical Communication 65, Working group 6, Geneva. IEC, 2005.

[6] R. Lewis, *Modelling control systems using IEC 61499: applying function blocks to distributed systems*, Institution of Electrical Eng, 2001.

[7] Christensen, J., Sünder, C., Zoilt, A., Colla, M., Strasser, T. *Execution Models for the IEC61499 elements Composite Function Block and Subapplication*. 5th IEEE International Conference on Industrial Informatics, pp. 1169-1175. Vienna, Austria. 2007

[8] Thramboulidis, K. *An Architecture to Extend the IEC61499 Model for Distributed Control Applications*, 7th International Conference on Automation Technology. Taiwan 2003.

[9] Vyatkin, V. *The potential impact of the IEC61499 standard on the progress of distributed intelligent automation*. International Journal of Manufacturing Technology and Management. ISSN 1368-2148, Vol. 8, no. 1-3, pp. 107 – 125. 2006

[10] OMG, Object Management Group, *Data Distribution Service for Real-time Systems, v1.2*, June, 2007.

[11] OMG, Object Management Group, "Common Object Request Broker Architecture: Core Specification", Version, 3.0.3, March 2004.

[12] OMG, Object Management Group, "Notification Service Specification", Version, 1.1, October 2004.

[13] Perez, F., Orive D., Marcos M., Estévez E., Morán G., Calvo I. *Access to Process Data with OPC-DA using IEC61499 Service Interface Function Blocks*, 14th IEEE International Conference ETFA2009, Palma de Mallorca, Spain, September 2009.

[14] CORBA, Object Management Group, "Notification Service Specification", Version, 1.1, October 2004.

[15] Ryll, M., Ratchev, S., "Application of the Data Distribution Service for Flexible Manufacturing Automation", Proceedings of World Academy of Science, Engineering and Technology, vol 31 July 2008, pp. 178-185

[16] http://www.opensplice.com/