Int J Adv Manuf Technol (2004) 24: 24–31
DOI 10.1007/s00170-003-1698-7

ORIGINAL ARTICLE

Kleanthis C. Thramboulidis · Chris S. Tranoris

# Developing a CASE tool for distributed control applications

Received: 21 December 2002 / Accepted: 13 March 2003 / Published online: 12 May 2004


**Abstract** The function block (FB) concept has been adopted by recent International Electrotechnical Commission (IEC) standards to define a methodology for the development of modular, re-usable, open, and vendor-independent distributed control applications. Control engineers are already familiar with the FB construct, and field devices and fieldbuses are expected to be compliant with this approach in the near future. New generation FB-oriented CASE tools are required to support the whole development process. This paper presents an approach to the design and development of an IEC-compliant CASE tool (ICT). The proposed approach is based on a four-layer architecture that successfully unifies the FB concept with the Unified Modelling Language. During the development of our prototype ICT, this architecture proved to be very significant for the identification of the key abstractions that the ICT must provide as building blocks of its various diagrams used during the modelling process of control systems.

**Keywords** IPMCS CASE tool · Distributed control applications · Function block · Engineering support system (ESS) · IEC 61499

## 1 Introduction

During recent years, due to the absence of standardization, a lot of fieldbus types have been installed in industry. Each fieldbus type has its own proprietary tool for the development and configuration of control applications. In addition, these tools do not exploit the new challenging technologies of software engineering – they address dimensions such as modularity, flexibility, extensibility, reusability, and interoperability very little or not at all . As a result, control applications are usually developed in the form of large monolithic software packages that are difficult to be maintained, modified, and extended [1]. Moreover, the interconnection of the resulting control applications on different fieldbus segments is addressed only after development, and this is performed using proprietary gateways. These gateways do not usually guarantee the satisfaction of the real-time requirements imposed by control applications and they do not support the unified development and configuration of the whole control system.

In today's competitive and ever-changing market, there is an always growing need for a more coherent approach in the development of control applications. Among the proposals that try to address this requirement we distinguish:

1. *The IEC 61499 standard*. This International Electrotechnical Commission (IEC) standard defines the basic concepts and describes a methodology to be used by system designers to produce modular and interoperable software for distributed industrial process control [2]. Although many researchers are already working on different aspects of the IEC proposal (e.g. [3, 4, 5]), the absence of tools and products that are compliant with this approach is evident. The function block development kit (FBDK) [6], which is the only known tool supporting this approach, allows the definition of FB types and the design of FB diagrams and produces the corresponding XML specifications. However, FBDK does not address major issues such as requirements specification, assignment and downloading of FB diagrams in field devices and fieldbuses, which make the tool, in its current version, inappropriate for the development of actual control systems.
2. *The Interface for Distributed Automation (IDA) architecture*. This architecture, which is proposed by

K. C. Thramboulidis (✉)
Electrical and Computer Engineering,
University of Patras, 26500 Patras, Greece
E-mail: thrambo@ee.upatras.gr
Tel.: +32-61-0997325
Fax: +32-61-0997316

C. S. Tranoris
Electrical and Computer Engineering,
University of Patras, 26500 Patras, Greece

the IDA group [7], attempts to establish a vendor-independent and open standard, adopting as its communication platform the TCP/IP protocol that is extended to support real-time communication. IDA's application model, although based on the reference architecture defined in IEC 61499, restrains the model in some aspects and extends or modifies it in others [8].

3. *The PROFInet engineering model*. This is the proposal of the Profibus User Organization developed to achieve open distributed automation. It adopts Ethernet, TCP/IP, and distributed component object model (DCOM) to provide a component-based development process [9]. Although the PROFInet approach is not IEC-compliant, an IEC-compliant mapping is planned for the near future.

This paper describes a systematic approach to the design and development of new generation computer aided software engineering (CASE) tools that are required to support the whole life cycle of FB-based IEC-compliant industrial process measurement and control systems (IPMCSs). Such an IPMCS CASE Tool (ICT) must support the engineer in the analysis, design, implementation, and configuration phases, as is very well described by the IEC 61499 standard. However, while working with the IEC standard for the development of such an ICT, we found that this standard does not fully specify: (a) the development process, for example the way that requirements are captured and translated to function block design diagrams; and (b) low-level communication services that are required to distribute, configure, and control the operation of FB-based control applications. We also found that a number of modifications are required to improve the IEC development process in terms of reliability, development time, and degree of automation. To address these issues, we defined:

1. A process for the development of distributed IPMCS applications
2. The CORFU framework, to fulfil the requirements imposed by the above process from the fieldbus, field device, and the communication subsystem levels
3. A four-layer IPMCS architecture, to exploit current software engineering practices through the IEC 61499 modelThese actions have resulted in a number of artifacts that constitute the infrastructure required for the development of our ICT prototype.

The remainder of this paper is organized as follows: Section 2 introduces the main concept of the FB approach. Section 3 outlines the proposed approach for the development of distributed control applications. Section 4 briefly presents the CORFU framework. Section 5 presents the conceptual model of the proposed ICT. The four-layer architecture is presented and the means by which this architecture influences the design of the proposed ICT are discussed. Section 6 discusses implementation issues of the ICT prototype and section 7 concludes the work.

## 2 The function block approach

To address the need for modular and interoperable software, the IEC 61499 standard defines the modelling of control applications using the FB concept. The FB, as defined in [1] is "an abstraction mechanism that allows industrial algorithms to be encapsulated in a form that can be readily understood and applied by industrial engineers, who are not specialists in the implementation of complex algorithms." The FB consists of a *head* that is connected to the event flows and a *body* that is connected to the data flows, as shown in Fig. 1, where the IEC 61499 graphical representation is given. The functionality of an FB is provided by means of algorithms, which process inputs and internal data and generate output data. The relationships between events and algorithm executions are specified by a special kind of state transition diagram that is called execution control chart (ECC). FB instances, interconnected by event and data connections, constitute the design model of the control application. The control application interacts with the controlled manufacturing process through a set of inputs and outputs, as shown in Fig. 2.

## 3 Our approach for the development of IPMCSs

To ameliorate the development process of distributed control applications, we adopted the best practices from component-based development, object technology, and the Unified Modelling Language (UML) [10]. The development process has been defined as a series of workflows, as described in detail in [11]. For the "capture requirements," the first workflow, we have adopted the well-accepted use-case concept introduced by Ivar Jacobson [12]. During this workflow, control and field engineers properly define the use cases of the system, i.e. the responses of the system to external events that originate either from devices or humans. During the next

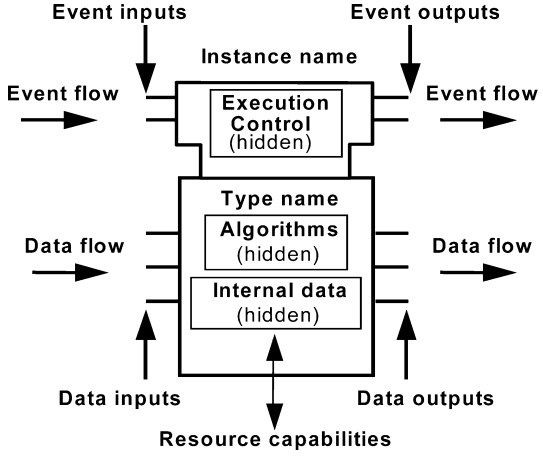



**Fig. 1** The IEC 61499 representation of the function block construct

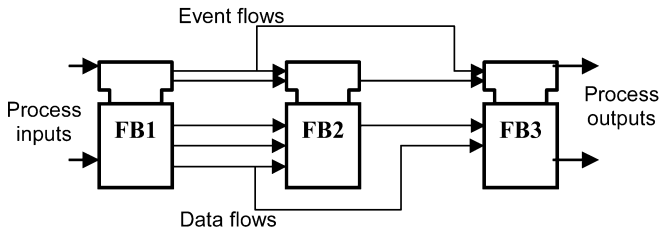



**Fig. 2** Function block instances are interconnected to form control applications

workflow, namely the "capture behaviour," engineers cope with the examination of the dynamic behaviour of the system. Object interaction diagrams are considered to be the first realization of system's use cases and are used to show the system's internal objects and the way they collaborate to provide the required behaviour. During the subsequent "capture static view" workflow, engineers deal with the design of the static view of the system in terms of class diagrams. Since the diagrams produced through the above process must be consistent, the engineer has to go back and forth through the workflows in order to better specify the analysis and early design models of the system. As soon as the above models have been defined, the engineer is ready to move to FB design diagrams. A set of transformation rules were defined to automatically transform the UML-based system model to an FB-based design model that is better understood by control engineers. "Refinement and evaluation," "model-verification," and "FB-distribution" are among the main workflows that complement the development process.

## 4 The CORFU framework

For our approach to be effectively applied, a number of requirements must be satisfied by the fieldbus, field device, and communication subsystem levels. The CORFU framework was defined to fulfil these requirements and to provide the infrastructure that is required for the development of the proposed ICT [13]. To satisfy the real-time constraints, imposed by the nature of IPMCS applications that must be distributed on a variety of fieldbus segments, the network topology shown in Fig. 3 was adopted. This topology utilizes ATM or fast switched-Ethernet or Gigabit Ethernet to obtain the required quality of service for the virtual fieldbus (VFB)-to-VFB communication channel. The VFB layer is used to abstract any commercial fieldbus to the IEC 61499 level, so the interoperability in the fieldbus level can be achieved. The industrial process-control protocol (IPCP) has been defined on top of TCP-UDP to provide the set of services required for the interaction between the ICT, the VFBs, and the human–machine interface (HMI) modules.

To enable the interworking unit (IU) to interconnect the many different commercial fieldbuses to the CORFU framework, a modular and highly expandable architecture has been defined [14]. According to this architecture, the implementation of the VFB is achieved through a wrapper layer that is defined to adapt the VFB to the specific fieldbus API. RTLinux [15] was selected to be the RTOS of the CORFU IU (CIU) and a prototype CIU was implemented for the Profibus case study.

## 5 Conceptual model of the ICT

### 5.1 Challenges and requirements

The second part of the IEC 61499 standard defines a methodology to be used by system designers to construct distributed control systems in terms of logically connected function blocks that run on different processing resources. New generation FB-oriented tools are required to support the whole life cycle of such systems. Using these tools, the engineers must be able to start with the analysis of the manufacturing process diagrams

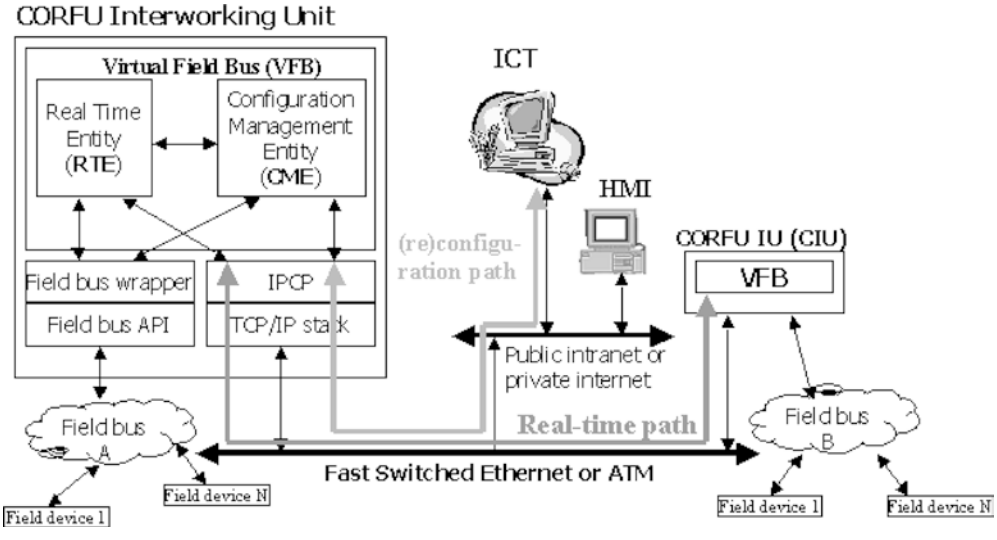



**Fig. 3** Network topology for interconnecting different fieldbus segments through the CORFU framework

in order to capture the control requirements. They should then be able to define the major areas of functionality and their interaction with the manufacturing process, as well as to exploit function blocks provided by intelligent field devices (e.g. smart valves), and to assign functionality into physical resources (e.g. programmable logic controllers, (PLCs), instruments, and controllers). All of the above should be accomplished independently of the underlying communication subsystem even in the most extreme case, where this subsystem is an aggregation of interconnected independent fieldbus segments from different vendors. To address these requirements, a qualified ICT should enable the engineer to:

- Create the analysis model of the control application
- Move into design by translating the UML-based analysis model to FB-based design diagrams
- Evaluate and improve the design
- Verify the application
- Distribute the application's components
- Download the application's components
- Develop configuration plans
- Supervise the operation of the resulting IPMCS system
- Prepare reconfiguration plans

## 5.2 The four-layer IPMCS architecture

To fulfil the above requirements and effectively utilize the CORFU framework in the design process of the proposed ICT, we refined the four-layer CORFU architecture (4LCA), which was first defined in [14], and enhanced it to cover the main phases of the development process. In this section, we primarily focus on this ''enhanced'' 4LCA and discuss how this architecture influences the design of our ICT. As shown in Fig. 4, the ''enhanced'' 4LCA consists of the following four layers:

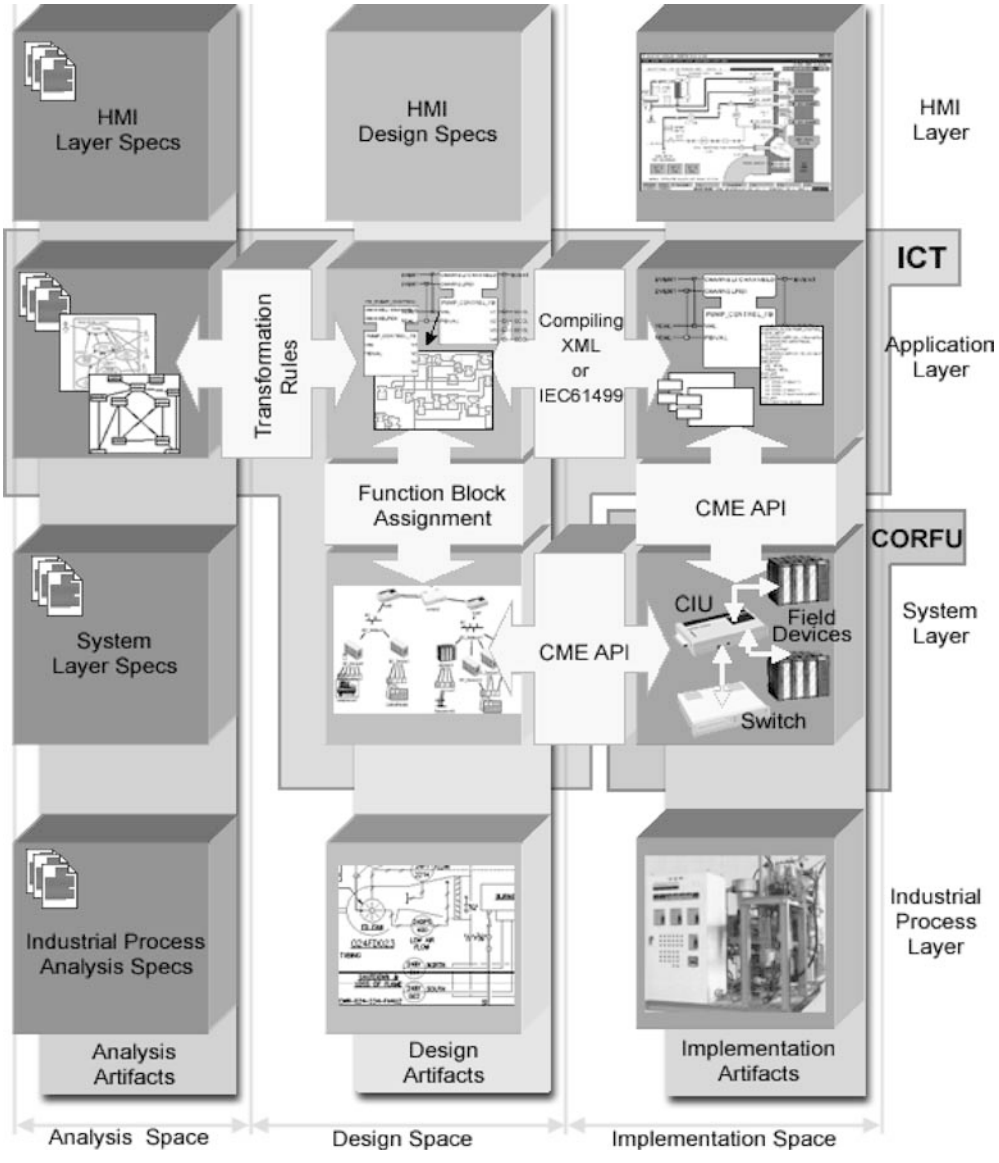



**Fig. 4** The proposed 4-layer architecture for the development of distributed IPMCSs

1. The *industrial process layer*. Boilers, valves, motors, etc. are the actual real-world plant components that constitute the implementation artifacts of this layer. The design representations of the real-world artifacts compose the design diagram of the controlled manufacturing process, which is usually expressed in the form of pipe and instrumentation diagrams. These diagrams, if defined in XML, could be imported by the ICT and properly used during the development process of the IPMCS.
2. The *system layer*. This layer includes the analysis, design, and implementation artifacts of the system on which the CORFU framework is implemented. Implementation artifacts include the fieldbus segments, the field devices, the IUs, and the network used for the real-time interconnection of fieldbus segments. The design artifacts of this layer include the abstractions used by the ICT to support the assignment and distribution of the application's components. These abstractions play the role of proxies of the actual real-world objects in the developer's workspace.
3. The *application layer*. This layer includes the required software constructs used in the analysis, design, and implementation of IPMCS applications. Use case diagrams, object-interaction and scenario diagrams, class, and FB diagrams are the most important diagrams supported in this layer. Among the most important artifacts of this layer are: FB type, FB instance, FB2FB data and event connection, and IPP2FB data and event connection. FB types in machine-readable representation and implementations of event and data connections are examples of implementation artifacts of this layer.
4. The *HMI layer*. This layer, which captures all the abstractions that concern the development and operational phases of HMI modules, is already supported by open and vendor-independent commercially available tools. However, the interaction with these tools must be considered.

Motivated by the above architectural perspective and in our attempt to fulfil the requirements of the ICT, we decided:

1. To fully support the analysis space of the application layer with a commercially available CASE tool
2. To automate the transition from analysis to design and to support it with the ICT
3. To fully support the design spaces of system and application layers with the ICT
4. To partially support (in the application layer) the translation from the design to the implementation model using commercially available code generation tools.

Through the above process, the 4LCA architecture allows for a smooth integration of our development process with the CORFU framework. It promotes re-usability, obtains interoperability in the fieldbus level, and allows the seamless development of distributed control applications. The proposed architecture proved to be very important for the identification of the key abstractions that the ICT must provide as building blocks of its various diagrams, which are used during the

**Fig. 5** The system layer design diagram

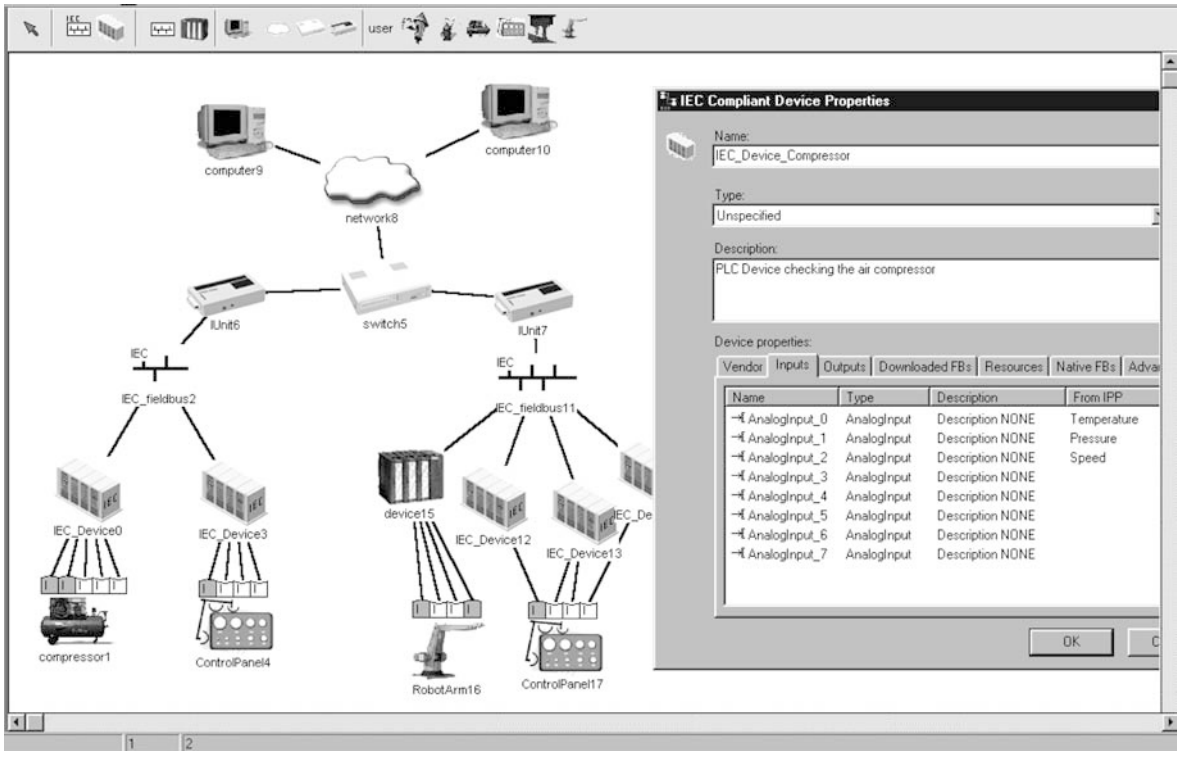


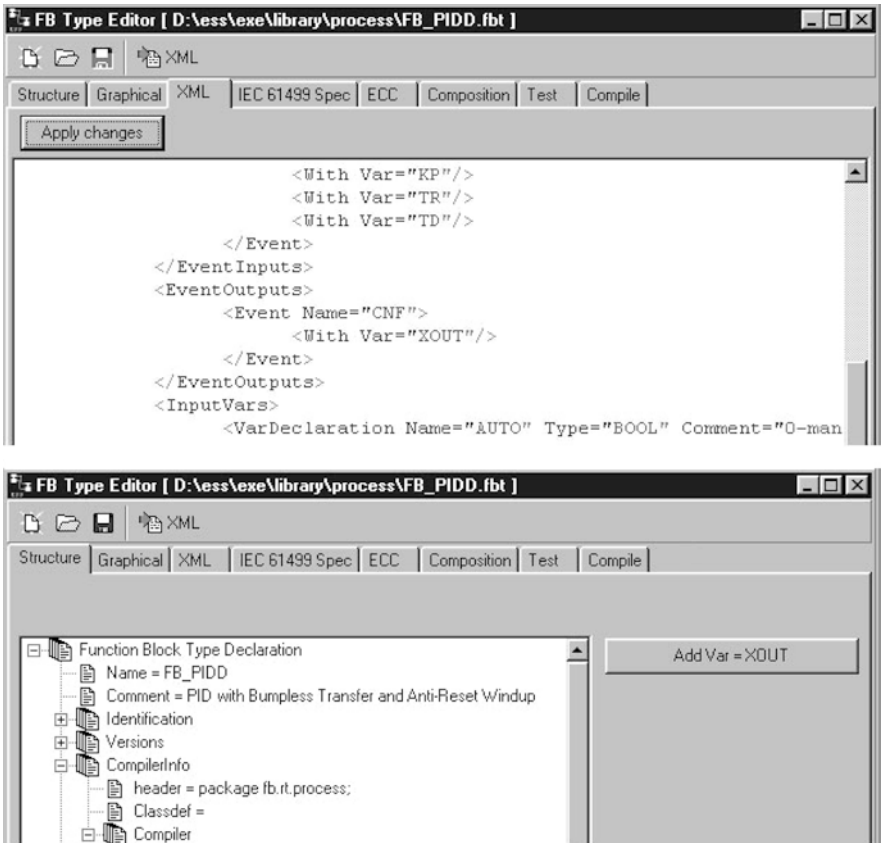

**Fig. 6** Properties of a sample industrial process terminator

modelling process of the system. As noted above, the 4LCA has been exploited for the design of a prototype ICT. This ICT currently supports the whole application layer and part of the system layer. Figure 5 shows the system layer editor, where the system design of the IP-MCS is constructed. For each one of the constructs of the system design diagram there is a corresponding analysis window to specify its properties. From the device analysis window, also shown in Fig. 5, the engineer has access to the properties of the device, which are the device's native and downloaded FBs, its resources, its inputs and outputs, and the industrial process parameters (IPPs) (e.g. temperature, pressure) to which inputs and outputs of the device are connected. Device type specific properties and the loading of the device specification are also supported.

The industrial process terminator (IPT) construct is used to represent, in the design space, the industrial process entities that are monitored or controlled by the application. IPTs are inserted into the design space of the system layer to properly define the interaction of the control system with the plant. Figure 6 shows the properties of the ''compressor'' IPT presented in the system diagram of Fig. 5. Each IPT has a number of IPPs, which are the inputs and outputs of the control application.

A snapshot of the FB editor is given in Fig. 7, where the tree-structure and the XML specification of an FB type are shown. Both textual and graphical IEC specifications are also supported. An ECC-editor allows the engineer to construct the execution control chart of the FB type. Options such as ''Test'', ''Compile'', ''FB composition'' complement the functionality of the FB editor.

**Fig. 7** The FB editor allows the construction of new FB types



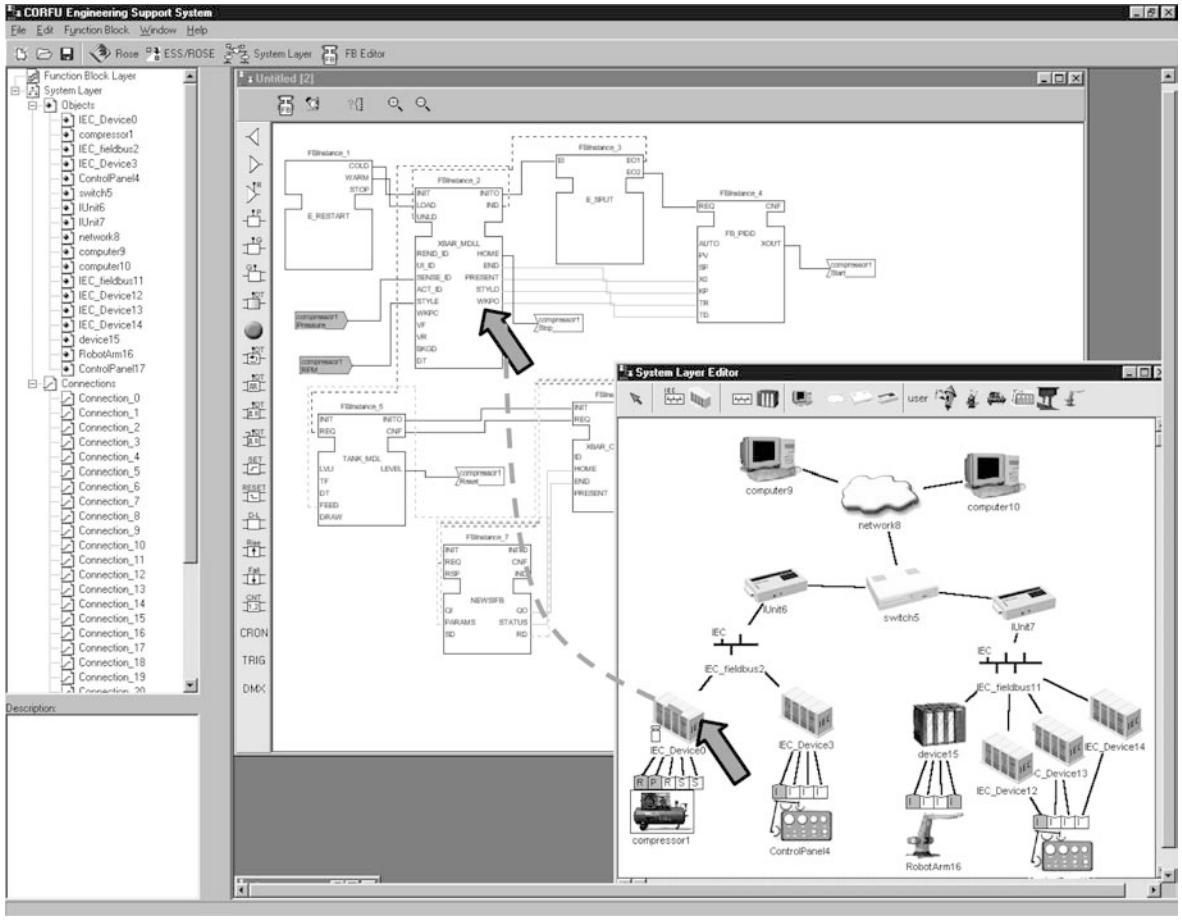



**Fig. 8** Function block and system design diagrams are used in the FB assignment process

FB instances of the defined FB types are used to create an FB design diagram of the application, as is shown in Fig. 8. The icons on the left sidebar of the FB editor are provided to allow the control engineer to implement in the model the different event scenarios that are supported by the IEC standard through the set of event function blocks that the standard defines. Such FBs are: ''event splitter'', ''event merger'', ''event rendezvous'', etc. These FBs are not supported by the proposed ICT; instead, an event-API has been defined that should be provided by an IEC-compliant device. The ICT automatically configures the device through the configuration event-API to provide the required behaviour during the operational phase. This approach results in simplified FB diagrams and improves the performance of the corresponding control application.

The ICT supports the FB assignment; it provides all the functionality required for the physical distribution of the software building blocks to the system layer components and mainly to field devices and fieldbuses. For example, for the assignment of an FB instance from the design space of the application layer to the target field device in the real-world layer, the control engineer has only to ''click'' on the FB instance and ''drag and drop'' it on the icon of the corresponding device in the design space of the system layer (Fig. 8). After this assignment, the downloading of the application to the target system in the real-world space of the system layer is a fully automated process. Actions of the ICT that have no meaning in the application design phase but refer to the configuration of the underlying framework are properly hidden by encryption and encapsulation mechanisms. The ICT, using the API of the configuration management entity (CME) of the VFB (Fig. 3), downloads to the device, through the IPCP and the proxy of the device, the machine-readable FB type definition that constitutes the atomic unit of distribution. The ICT then issues a ''new instance'' command to create the software module which implements the FB instance in the target device (in the implementation space of the system layer).

## 6 Developing the prototype ICT

Concerning the development of the prototype ICT, starting from scratch was considered to be a waste of time. Modern commercially available CASE tools that support the UML notation can be used instead to elaborate to modern IEC-compliant ICTs. We selected Rational Rose [16], which is one of the most popular CASE tools that support the whole life cycle from requirements through the final implementation.

However, any other CASE tool that supports the OO approach may be used. We decided to develop an external tool to support the extra functionality that is required by the ICT and is not provided by the general-purpose CASE tool. The interaction of our tool with the general-purpose CASE tool is based on communication mechanisms that are usually provided by this kind of tools. Rose, in our case, supports two such mechanisms: the first one is based on a custom scripting language, while the second is based on COM automation. Rose is used, through its type library, as a COM server to get type information about each object of the system model. This information refers to the interfaces of each object, the member functions of each interface, and the arguments of the member functions. Borland's Delphi integrated development environment (IDE), which was used for the development of our prototype ICT, provides a type library editor and a tool that translates a type library file to Delphi constructs. The IDE also creates the equivalent dispatch interfaces through which these constructs can be handled. More details on implementation issues can be found in [17]. An evaluation version of the prototype ICT can be downloaded from http://seg.ee.upatras.gr/corfu.

## 7 Conclusions

In this paper, an approach for the development of FB-oriented, IEC 61499-compliant engineering support systems has been proposed. The primary objective of this work was to simplify the development of distributed control applications by hiding complexities associated with fieldbuses and field devices. Although we selected the IEC 61499 standard as the base of our work, we found that this standard does not address all of the issues involved in such an undertaking. We were guided to propose a number of modifications and extensions to ameliorate the development process. We adopted current software engineering practices and focussed on the abstractions required to identify reusable components and the constructs necessary to support customisation in distributed control applications. A four-layer architecture was constructed, taking into account issues such as modularity, expandability, robustness, and flexibility. The introduced architecture has proven to be very helpful in the development process of a prototype ICT; therefore, the proposed approach and the prototype developed should provide a generalized framework and a set of development guidelines for the next generation of ICTs that are necessary for the development of FB-based distributed control systems.

**Acknowledgements** This work was supported in part by the Greek General Secretariat for Research and Technology.

## References

1. Lewis R (2001) Modelling control systems using IEC 61499. The Institution of Electrical Engineers
2. IEC Technical Committee TC65/WG6 (2000) IEC 61499 Industrial-process measurement and control specification. IEC Draft
3. Vyatkin V, Hanisch HM (2001) Formal-modelling and verification in the software engineering framework of IEC 61499: a way to self-verifying systems. In: Proceedings of the IEEE conference on emerging technologies in factory automation (ETFA '01), Nice, 15–18 October, pp 113–118
4. Xu Y, Brennan R, Norrie X (2001) Agents, holons and function blocks: distributed intelligent control in manufacturing. J Appl Syst Stud (Special Issue on Industrial Applications of Multi-Agent and Holonic Systems) 2(1):1–19
5. Fletcher M, Norrie DH (2001) Real-time reconfiguration using an IEC 61499 operating system. In: Proceedings, 15th international parallel and distributed processing symposium (IPDPS-01), San Francisco, CA, 23–27 April 2001
6. Function block development kit, Rockwell Automation, http://www.holobloc.com. Cited 22 Nov 2002
7. IDA – the internet of automation technology (2001) White paper, v 1.0, IDA-Group, http://www.ida-group.org. Cited 15 Oct 2002
8. IDA – interface for distributed automation: architecture description and specification (2001) Revision 1.0, IDA group, November 2001
9. PROFInet – architecture description and specification (draft) (2001) PROFIBUS International, http://www.profibus.com/technology/ documentation /texte/01295/. Cited 9 Oct 2002
10. Rumbaugh J, Jacobson I, Booch G (1999) The UML reference manual. Addison-Wesley, Reading, MA
11. Tranoris C, Thramboulidis K (2002) From requirements to function block diagrams: a new approach for the design of industrial control applications. In: Proceedings, 10th Mediterranean conference on control and automation, (MED2002), Lisbon, Portugal, 9–12 July 2002
12. Jacobson I (1992) Object-oriented software engineering: a use-case driven approach. Addison-Wesley, Reading, MA
13. Thramboulidis K (2002) Development of distributed industrial control applications: the CORFU framework. In: Proceedings, 4th IEEE international workshop on factory communication systems, (WFCS '02), Mälardalen University, Västerås, Sweden, 27–30 August 2002
14. Thramboulidis K, Tranoris C (2001) An architecture for the development of function block oriented engineering support systems. In: Proceedings, 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (IEEE CIRA 2001), Banff, Canada, 29 July–1 August 2001
15. Yodaiken V, The RTLinux manifesto, http://www.rtlinux.com/archive/rtlmanifesto.pdf. Cited 7 May 2001
16. Rational Rose, Model-driven development with UML, http://www.rational.com/media/products/rose /D185G_Rose.pdf. Cited 9 Sept 2002
17. Tranoris C, Thramboulidis K (2002) A UML based engineering support system for the development of distributed control applications. In: Proceedings, 4th international workshop on computer science and information technologies, Patras, Greece, 18–20 September 2002