Using UML for the Development of Distributed Industrial Process Measurement and Control Systems

Kleanthis C. Thramboulidis

Electrical & Computer Engineering Department, University of Patras,
265 00 Patras, Greece, thrambo@ee.upatras.gr

***Abstract*-- Software industry increasingly faces today the challenge of creating complex custom-made Industrial Process Measurement and Control Systems (IPMCSs) within time and budget, while high competition forces prices down. A lot of proprietary solutions address the engineering process and evolving standards exploit the function block construct as the main building block for the development of IPMCSs. However, existing approaches are procedural-like and they do not exploit the maximum benefits introduced by the object technology. In the context of this work, new technologies in Software Engineering as well as modern CASE tools, which assist to improve the efficiency of software development process, are considered. The Unified Modeling Language (UML) was adopted for the definition of a notation to assist in the design and development of open distributed IPMCSs. The proposed notation constitutes the heart of our object-oriented framework that attempts to improve the engineering process in terms of reliability, development time and degree of automation.**

***Index Terms*-- UML, Industrial Process Measurement and Control Systems, engineering support system, fieldbus, Object-Oriented design.**

I. INTRODUCTION

The many different types of commercial fieldbuses resulted in a wide diversity of devices and tools to support the development of Industrial Process Measurement and Control Systems (IPMCSs). A number of different approaches try to provide interoperability in device and fieldbus level. MMS, an application layer protocol of the ISO/OSI reference model, was considered to be the most important of them. Even though MMS is used in environments in which timing constraints are fundamental, it deals poorly with timing specifications [1]. MMS extensions to handle the real time constrains were never adopted by the market. More recent standards like IEC 61131 and IEC 61499 have introduced the function block construct as the main building block, to facilitate the development of IPMCSs [2]. The function block, is defined according to IEC 61499, as "a functional unit of software, comprising an individual, named copy of the data structure specified by the function block type, which persists from one invocation of the function block to the next. The functionality of the function block is provided by means of algorithms, which process inputs and internal data and generate output data". The above standards attempt to address the interoperability issue, but they still have a long way to go. Over and above, the function block approach is procedural-like and it does not exploit the maximum benefits introduced by the Object Technology (OT) [3].

OT is being recognized as perhaps the best-known way to revolutionize and improve dramatically the efficiency of the software development process. OT is attracting a major area of interest from industrial software developers in recent years, under the promise to address all the concepts of modern software engineering or to make the introduction of new such concepts, where appropriate. Information hiding, messaging, data abstraction, encapsulation above the subprogram level, polymorphism and concurrency are examples of such concepts. OO analysis produces models that are easier understood by end users, as they are direct representations of real world concepts. The OO paradigm promotes and facilitates reusability, interoperability and, when applied well, produces solutions, which closely resemble the original problem. The use of the OT leads up to systems that are easily modified, extended and maintained.

Software industry, on the other hand, increasingly faces today the challenge of creating complex custom-made distributed control systems within time and budget, while high competition forces prices down. New technologies in Software Engineering as well as modern CASE tools that assist in improving the efficiency of software development process should be considered for the development of distributed IPMCSs.

All the above guided us, to consider the applicability of the Unified Modeling Language (UML) in the development process of distributed IPMCSs. UML has emerged as the software industry's dominant language and is already an Object Management Group (OMG) standard. It represents a collection of best engineering practices that have been proved successful in the modeling of large and complex systems. OMG is proposing the UML specification for international standardization for information technology [4]. Wide recognition and acceptance, which typically enlarge the market for products based on it, will be the major benefits.

In the context of this work, we made an attempt to tailor UML to the particular requirements of distributed IPMCSs. We have exploited both a lightweight and a heavyweight extension mechanism, to create a set of constructs to assist the engineer in the design and development process. Although UML currently provides only lightweight extension mechanisms, it is expected that heavyweight ones will also be supported in the near future

[4]. We consider the proposed notation as an important feature of our framework based approach in the development of distributed IPMCSs.

Frameworks provide an important enabling technology for reusing both the architecture and the functionality of software components allowing the IPMCS engineer to ignore all the complexities inherent in field devices and fieldbuses and to concentrate on the actual problem to be solved. Our framework based approach assists process and system engineers in the development and configuration of distributed IPMCSs. The hole engineering process would be improved in terms of reliability, development time and degree of automation compared with the one proposed by the IEC 61499.

The rest of this paper is organized as follows. In section II, we briefly describe our framework-based approach and highlight the adopted technologies. A reference is made to our interworking unit that forms the framework's infrastructure. In section III, the notation that is one of the most important features of our framework, is presented in terms of UML's extension mechanisms. The requirements specification of an Engineering Support System (ESS) that supports the notation is briefly discussed. Finally, the last section is used to conclude the paper.

II. Our Framework Based Approach

A. The Framework

A lot of proprietary solutions address the engineering process of the IPMCS. Evolving standards like IEC 61499 and the more recent IEC61804 introduce the function block construct as the main building block for the development of IPMCSs. However the whole approach is procedural-like and it does not exploit the maximum benefits introduced by the object technology. To exploit the many benefits of the object technology as they were encountered in our many projects where we applied it, we were guided to a framework-based approach. Figure 1 shows a high level view of our IMPCS framework. The framework address both the IPMCS engineering process, which utilizes the fieldbus-to-ESS channel and the IPMCS operational phase, which utilizes the fieldbus-to-SCADA client channel and the fieldbus-to-fieldbus channel.

The framework consists of a set of collaborating object classes that embody an abstract design capable to provide solutions for the family of IPMCS's related problems. It increases the reusability in both architecture and functionality by addressing issues such as interoperability and integrated development of distributed IPMCSs. Since an important issue for the success of a framework is its ability to support the integration of legacy systems and enterprise applications, an attempt was made to this direction. The framework may be considered in terms of:

1. An interworking unit, that is the infrastructure for the development of the interoperability mechanism.
2. The specification of a virtual fieldbus.
3. An Engineering Support System, that is required to support, according to IEC 61499, the design, implementation, commissioning and operation of IPMCSs.
4. A function block compliant extension of the UML notation to assist in the design and implementation of IPMCSs applications.

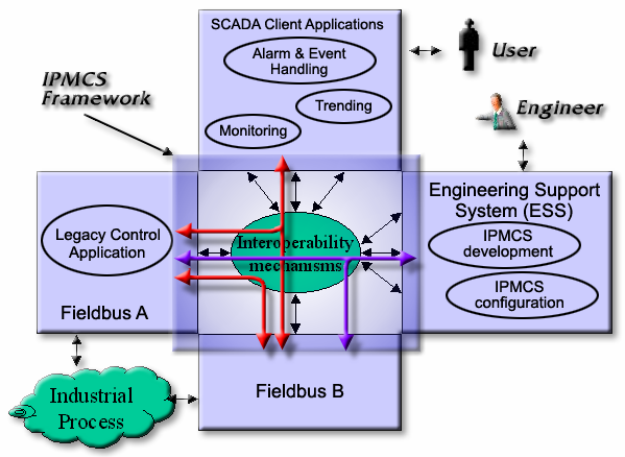

Figure 1. High-level view of the IPMCS framework.

For the fieldbus-to-SCADA client channel, we adopted Internet although its role as a channel over which SCADA applications are built is a fairly recent phenomenon. Internet's impact has been substantially greater than that of other proprietary channels in existence for several decades. For the same reasons we adopted Internet for the fieldbus-to-ESS channel. However Ethernet in its current status is not possible to be considered for the fieldbus-to-fieldbus channel. Alternatives such as ATM, Gigabit Ethernet and Fast switched Ethernet may be successfully adopted.

B. Network Topology

Figure 2 presents the network topology we selected to meet the real-time constraints. An interworking unit is used to interconnect each fieldbus to the backbone network communication subsystem. This communication subsystem must provide the quality of service (QoS) required to meet the timing requirements. ATM is one of the successfully used technologies for the interconnection of fieldbuses [5][6]. However, we have also under consideration Gigabit Ethernet or switched Fast Ethernet, which seems to meet the needs of the framework in a lower cost.

C. Interworking Unit

Due to the strict timing constraints imposed by the application domain, the interworking unit is characterized as hard real-time. Data processing is expected to recognize and to react to events as soon as possible or even in the ideal case instantaneously. The use of an RTOS was mandatory. We selected the RT Linux, which is an open source real time Linux implementation running Linux as its lowest priority execution thread. A robust and modular architecture for the interworking unit has been defined and

is currently under development [7][8]. Further, the interworking unit was structured in such a way so as to facilitate the integration of existing fieldbuses.

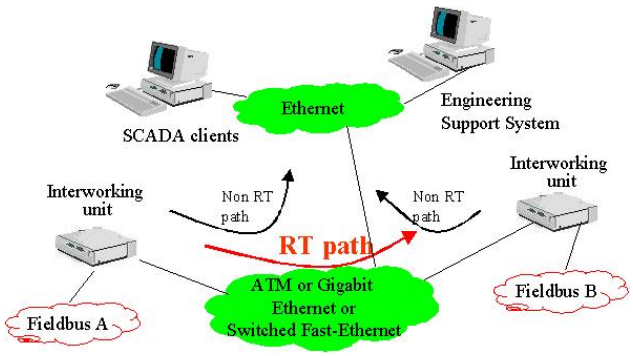

Figure 2. The proposed network topology.

Since the most of the commercial SCADA systems have adopted the de-facto market standard OLE for Process Control (OPC), we decided to provide an OPC compliant interface over the fieldbus-to-SCADA client channel [9]. This way, we can interface with no extra cost with the majority of commercial SCADA systems based on it.

From the perspective of developing independent engineering support systems, the VF may be considered as an aggregation of device proxies, industrial process parameters and application specific parameters. A formal device specification that enables device interoperability is indispensable for the definition of the device proxy. Several notations, known as Device Description Languages (DDLs), already support the specification of field devices. HART DDL, Profibus device description and Lonworks DDL are some examples. The problem is only partially addressed by evolving standards like IEC61499 and IEC61804. Using the related work on Profibus presented in [10], we are developing an Object-Oriented field device model compliant with the above standards to support the engineering tool in the design and instrumentation phase [11]. A lot of virtual fieldbus operations are derived from this model.

For the interworking unit to provide interoperability in the fieldbus level a Virtual Fieldbus (VF) has been defined. It exposes an object-oriented API to support:

1. the interconnection of fieldbus device islands of the various commercial fieldbuses,
2. the uniform interface to the enterprise intranet and
3. the development of fieldbus independent Engineering Support Systems

With the explosive growth of Internet, emerging standards such as XML make transmitting data over the Web inexpensive and efficient [12]. XML looks promising as the industry standard in the IPMCS domain. It was adopted in the context of the proposed framework to allow applications to communicate regardless of their programming model. It is expected that the majority of the fieldbus vendors will support XML as a universal Internet format since it address successfully the issues many earlier proprietary solutions tried to resolve. Part 2 of the IEC61499 standard, address the definition of a formal information model that will enable CASE tools and utilities to manipulate and exchange system designs based on function blocks. Customized tags in additions to those used by the IEC 61499, must be defined for the description of the represented information's structure so it is easy to transform structured industrial data into XML and vice versa.

III. THE PROPOSED NOTATION

A. *UML's extension mechanisms*

UML is a general-purpose visual modeling language that is used to specify, visualize and document the artifacts of software system [13]. It is becoming the de-facto standard and is currently supported by the majority of the design and development tools.

In our attempt to tailor this modeling language to the particular requirements of the IPMCS, we have exploited its inherent extension mechanisms, to create a set of constructs to assist the engineer in the design and development process. UML currently provides only lightweight extension mechanisms. However, it is expected that heavyweight ones will also be supported in the near future.

UML's lightweight extension mechanisms have been designed so that commercial CASE tools can store and manipulate the extensions without understanding their full semantics. The extensions are stored and manipulated as strings, so they can be entered, stored as part of a model and passed to other CASE tools. The particular syntax and semantics of these extensions must be defined so as to be the basis for the development of specific back-end tools or add-ins that would be written to process the introduced extensions. Constructs, which are defined using the extension mechanisms, reduce the semantic gap between the physical problem and its analysis and design models. Further, UML's inherent facility for interchanging models among several OO CASE tools and IDE's, will ease the design and development of distributed IPMCSs.

B. *IPMCS stereotypes*

We mainly utilized the UML's extension mechanism that is called stereotype. A stereotype is a kind of model element defined in the model itself. The stereotyped elements would have their own distinct icons since the most of the commercial CASE tools support this extension. We defined the following stereotypes as the main building blocks of the IPMCS model:

- the FunctionBlock-stereotype (FB-stereotype),
- the Data-Dependency-stereotype (DD-stereotype),
- the Control-Dependency-stereotype (CD-stereotype),
- the Industrial Process Terminator stereotype (IPT-stereotype), and
- the Real-World Object stereotype (RWO-stereotype).

The IPT-stereotype is used to represent any device of the industrial process that acts as source or sink of data or control information for the control application. It is discriminated to input-IPT, output-IPT and input/output-IPT. In the steam boiler system [14], for example, the steam quantity measuring unit, the pump control unit, the water level measuring unit and the valve control unit are all represented in the system's context diagram, by means of the IPT stereotype construct, as is shown in figure 3.

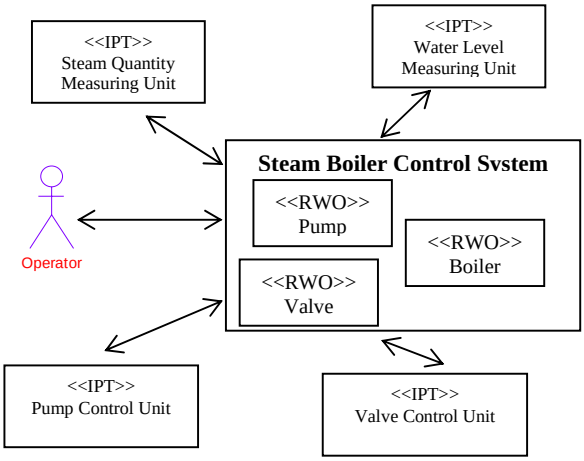

Figure 3. IPT stereotypes in system's context diagram.

For every real-world object, in the application domain, that is relevant to the problem, there should be a corresponding software object in the system [15]. For example, in the steam boiler system, the steam boiler, the valve, and the pumps, are all relevant real-world objects because they impact the steam boiler control system. The real-world relevant objects maybe considered as I/O devices that interface to the control system, providing inputs and receiving outputs from the system. As an example, the valve and the pumps are real-world objects that are equipped with actuators that receive outputs from the control system. On the other hand, the steam boiler is a real-world object that has sensors that provide inputs to the control system.

Service provider objects, either called control objects, provide the overall coordination for the group of objects that cooperate for the execution of each service of the control system. The whole system should be structured into subsystems, where each subsystem provides all the functionality provided by the objects it contains. Subsystems can be represented either as packages or as composite objects. We recommend the use of composite objects since the UML package has limited usage: it cannot participate in object interaction (collaboration) diagrams. In this case, the subsystem class diagram is used to depict the relationships between subsystems.

Later in the design phase the use of the wrapper stereotype provides a virtual interface that hides the actual interface to the real-world I/O device. Users of the wrapper stereotype are insulated from changes to the real-world device i.e. the change of a pump or a valve.

C. The function Block stereotype

The information content and form of the FB-stereotype are the same as those of a class but its meaning and usage is different. The FB-stereotype can be treated as special kind of class, it has attributes and operations, but it has special constraints on its relationships to other elements of the model as well as on its usage. There is no need for the introduced constraints to be automatically verified by a general-purpose CASE tool. They can be enforced manually or verified by an add-in tool that understands the specific stereotype. We use tagged values to store the additional properties of the FB-stereotype, which are not supported by the corresponding base element. A tagged value of a FB-stereotype is a pair of strings that stores a piece of information about the specific stereotype. The tag is, a name of some property that the modeler must record and the value given to this property for the specific instance. Tagged values would also provide a way to attach to the model elements, the implementation-dependent information, that is needed by code generators and other add-ins such as project planners and report generators.

In more detail the FB-stereotype is defined as a special kind of class that has:

- Attributes
- Methods
- Data-Dependencies
- Control-Dependencies
- A state-transition diagram or an execution control chart (ECC)

The FB-stereotype attributes represent either its local parameters or the internal parameters that correspond to its data-dependencies. The FB-stereotype methods are discriminated in:

- Set-Get methods. The set and get methods are automatically produced by the code generator add-in to implement the FB-stereotype's data-dependencies.
- Response methods. A response method defines the response of the FB-stereotype to a specific control-dependency.
- Common methods, which are mainly used to maximize reusability.

D. Data and control dependency stereotypes

The DD-stereotype is defined as a special kind of association mainly between FB-stereotypes and FB-stereotypes and IPT-stereotype. There is a DD-stereotype for each data input of the corresponding function block as is defined by IEC 61499. The resulting association is a producer-consumer data association. The semantics of the DD-stereotype are extended so as to represent the set of data inputs provided by the same function block.
For each DD there is an internal data parameter that takes its value from the appropriate set or get method that is automatically derived by the tool that generates the code to

implement the DD. For each DD the designer has to define which of the associated elements play the producer, consumer and actor roles. The engineering tool uses this information to produce the appropriate set or get methods that are used to implement the DD. In more detail the corresponding internal data parameter of a DD may take its value in one of the following two ways:

- The producer diagram element sends a message of type `set(attributeName, attributeValue)`
- The consumer diagram element sends a message of type `get(attributeName)` to the producer element which may be FB or IPT.

The CD-stereotype is defined as a special kind of association between FB-stereotypes and FB-stereotypes and IPTs. There is a CD-stereotype for each event input of the corresponding function block. For each CD, a corresponding method called response method, defines the behavior of the consumer FB to the event captured by the CD. In the context of the response method the FB may carry out a series of actions. These actions may:

- perform computational processes,
- modify the values of specific attributes and fire an ECC transition,
- enable or disable continuous methods,
- trigger one-shot methods,
- enable one or more DDs,
- enable one or more CDs

The need for other kind of actions shall be examined and the required mechanisms for their implementation must be defined.

To make a formal representation of the above constructs and to facilitate the implementation of related ESS, the UML notation was used. Figure 4 shows part of the derived UML model that clearly represents most of the above concepts.

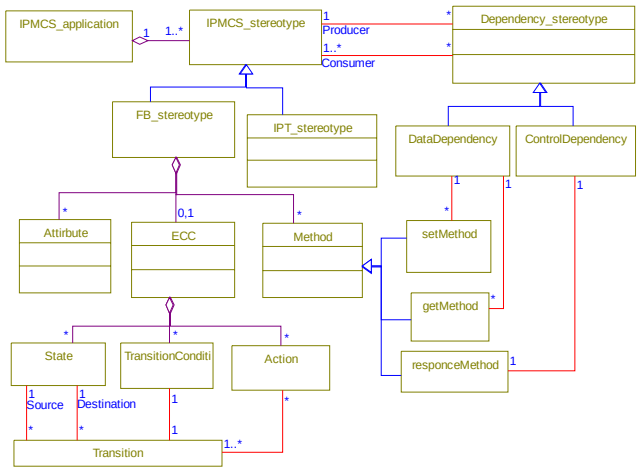

Figure 4. Part of the FB-stereotype's UML model

E. *The metamodel pattern*

An alternative way to represent the above constructs and to satisfy the need for IPMCSs models to be reliably stored, shared, manipulated and exchanged across tools, is to adopt the meta-model architectural pattern [16]. In general meta-modeling is recognized as one of the most powerful techniques for managing the complexity of distributed applications like the ones required by modern IPMCSs. The application of this architectural pattern, which is a proven infrastructure for defining the precise semantics required by complex software models, resulted in the three layer meta-model architecture that is shown in figure 5. The layer in the bottom represents the user objects layer that is an instance of the application's Analysis Model and is defined by use of the engineering tool. The model layer captures the specific applications analysis and design models. Specific design patterns for function block oriented systems, such as the Model-View-Controller presented in [17] would be captured in this layer. The metamodel layer is composed of a set of metaclasses that define the precise semantics required by the complexity of the IPMCS models for their reliable storage, sharing, manipulation and exchange across engineering support systems. Dependency, data and control dependency, and Function block are examples of metaclasses of this layer.

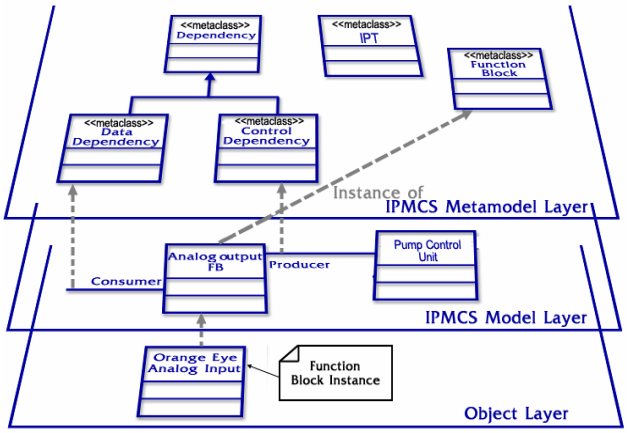

Figure 5. The 3-layer meta-model for the development of IPMCSs.

F. *ESS's requirements specification*

The use case driven approach has been used for the requirements specification of ESS. At the same time an attempt has been made to be compatible with part 1 of the IEC61499, which address the rules for the declaration and use of the function blocks and the requirements for compliant systems and standards. According to this, an ESS must support the engineer, in both the analysis and design phases as well as in the implementation phase. Using such a system, the engineer must be able to start with the analysis of the plant diagram so as to capture the control requirements. Then, he should be able to define the major areas of functionality and their interaction with the plant. During this task, he should be able, to exploit FB-stereotypes provided by intelligent field devices such as

smart valves, but also to assign functionality into physical resources such as PLCs, instruments and controllers. All the above should be accomplished independently of the underlying communication subsystem even in the extreme case, where it is an aggregation of interconnected fieldbus segments, from different vendors. During the design phase of the application, the engineer analyses the physical plant and uses FB-stereotypes, DDs, CDs, and other software constructs to represent the key abstractions of the application domain. He inserts the appropriate IPTs into the working space, so as to properly define the interaction of the control system with the plant. He also inserts field devices to the engineering workspace while connecting the field devices to the proper fieldbus. He makes use of device's specifications to create device representatives (device proxies) into the working space. Later during the design of the system, the engineer physically distributes the software building blocks to the available field devices and fieldbuses through their proxies. Partitioning, assignment, scheduling are among his/her most important tasks to properly prepare for downloading. Testing and monitoring the application are among the optional functionalities that an ESS must support.

Adopting either the stereotype approach or the metamodel one, existing CASE tools that support the UML notation may be used to elaborate to modern Engineering Support Systems. A great increase is encountered during the last years in the number and complexity of commercial CASE tools. Their functionality has been expanded and their degree of automation has been significantly improved. The most of the modern commercial CASE tools support the UML notation and a lot of this know how can be successfully utilized for the development of the IPMCS Engineering Support Systems. We adopted an open source CASE tool for the development of our ESS.

IV. CONCLUSIONS

The function block concept has been proposed by recent standards for the development of distributed IPMCSs. The function block approach is purely functional and does not exploit the benefits of the Object Technology. Object Orientation is now a mature technology with many remarkable commercial tools supporting the whole software life cycle. UML is becoming the de-facto standard and is currently adopted by the majority of modern CASE tools.

An extension of the UML notation was defined to enable the design and development of distributed IPMCSs using the widely accepted UML notation and the mature general-purpose CASE tools. The notation could be used as a modeling language to visualize, specify, construct and document the artifacts of IPMCSs. Both the stereotype extension mechanism and the metamodel one, were considered to capture the key abstractions required for the UML based development of distributed IPMCS applications. The FB stereotype, the Data and Event Dependency and the IPT stereotype are the most important constructs defined.

Our primary objective in defining our framework based approach was to simplify the development of IPMCS applications by hiding complexities associated with fieldbuses and field devices. We focused on the abstractions required to identify reusable components and on the constructs necessary to support customization of the framework required by the specific application.

We are currently working on the development of the interworking unit and the prototype of an Engineering Support System.

Acknowledgements

This work has been funded in part by the Greek General Secretariat for Research and Technology in the context of PENED 99 ED 469 project. I gratefully thank Chris Koulamas and Chris Tranoris, members of the development team, for their helpful discussions.

References

[1] Justin Akazan, Zoubir Mammeri, "Real-Time extensions to MMS", IEEE International Workshop on Factoy Communication Systems, 1995.

[2] IEC Technical Committee TC65/WG6, "IEC61499 Industrial-Process Measurement and Control – Specification", IEC Draft 2000

[3] R.W. Lewis, "Modeling Distributed Control Systems Using IEC61499 function block s", Technical Articles, URL: http://www.searcheng.co.uk/selection/control/tech.htm

[4] Cris Kobryn, "UML 2001: A standardization odyssey", Communications of the ACM, October 1999, vol.42, No 10.

[5] O. Kunert, " Interconnecting fieldbuses through ATM", IEEE international workshop on factory communication systems, 1997.

[6] C.Cseh, M.Jansen, J.Jasperneite, "ATM networks for factory communication", 7th IEEE International Conference on Emerging Technologies and Factory Automation, 1999.

[7] K. Thramboulidis, C. Tranoris, "An Architecture for the Development of Function Block Oriented Engineering Support Systems", IEEE International Symposium on Computational Intelligence in Robotics and Automation, Canada 2001.

[8] C. Tranoris, S. Aslanis, K. Thramboulidis, "Using RT_Linux for the interconnection of industrial fielsbuses" ASME international, First National Conference on Recent Advances in Mechanical Engineering, September 17-20, 2001 Patras, Greece.

[9] OPC Task Force, "OLE for Process Control Specification, Final Release V1.0", OPC Foundation 1998.

[10] Christian Diedrich, Peter Neumann, "Field Device Integration in DCS Engineering using a Device Model", IEEE 1998.

[11] K. Thramboulidis, A. Prayati, "Field Device Specification for the Development of Function Block Oriented Engineering Support Systems", submitted for presentation to ETFA 2001.

[12] Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, 6 October 2000.

[13] "OMG Unified Modeling Language Specification", Version 1.3, First Edition, Object Management Group Inc. March 2000.

[14] J. R. Abrial, "Steam-boiler control specification problem", August 10, 1994.

[15] K.Thrampoulidis, K.Agavanakis "Introducing Object Interaction Diagrams : A technique for A&D" Journal of Object-Oriented Programming (JOOP) June 1995.

[16] Maarten Boasson, "The Artistry of Software Architecture", IEEE Software, November 1995, vol. 12 No 6.

[17] J.H. Christensen , "Design patterns for systems engineering with IEC61499".