AN OBJECT-ORIENTED FRAMEWORK FOR THE DEVELOPMENT OF DISTRIBUTED INDUSTRIAL PROCESS MEASUREMENT AND CONTROL SYSTEMS

Kleanthis Thramboulidis, Chris Tranoris
Electrical & Computer Engineering Department, University of Patras, 265 00 Patras, Greece.
E-mail: thrambo@ee.upatras.gr, tranoris@ee.upatras.gr

Chris Koulamas
Industrial Systems Institute, 265 00 Patras, Greece.
E-mail: koulamas@ee.upatras.gr

Key words: IPMCS, Object-Oriented framework, engineering support system, fieldbus.

Abstract: Software industry increasingly faces today the challenge of creating complex custom-made Industrial Process Measurement and Control System (IPMCS) applications 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 paper, new technologies in Software Engineering that assist in improving the efficiency of software development process are considered. An Object-oriented framework is defined, to improve the engineering process of IPMCSs in terms of reliability, development time and degree of automation. This framework embodies an abstract design capable to provide solutions for the family of distributed IPMCSs. It will attempt to increase reusability in both architecture and functionality by addressing issues such as interoperability and integrated development of distributed IPMCSs.

1. INTRODUCTION

Today's rapidly changing market requirements impose the need of improving the agility of manufacturing systems. The always growing need for innovative products, forces manufacturing plants to improve their ability to quickly respond to market demands by designing competitive products and modifying existing ones. Until recently, most of the Industrial Process Measurement and Control Systems (IPMCSs) have been based either on traditional distributed control systems or on programmable logic controllers. In both cases, the systems are composed of monolithic applications that are almost impossible to integrate and even to expand. Modularity, flexibility, extensibility, reusability and interoperability are dimensions slightingly addressed by many traditional proprietary engineering tools and products. Even more, the most of the traditional products and tools are far away from the new challenging technologies in Software Engineering. The IPMCS software industry increasingly faces today, the challenge of creating complex custom-made distributed control systems within time and budget, while high competition forces prices down.

On the other hand, evolving standards, like (IEC61499, 2000) and the more recent (IEC1804, 1999), define the basic concepts and a methodology for the design of modular, re-usable, distributed industrial process, measurement and control systems. They define, the function block construct as the main building block of IPMCS applications, in a format that is independent of implementation. They also define, the way that function blocks can be used to define robust, re-usable software components that constitute the distributed IPMCSs, and a methodology to be used by system designers to construct distributed control systems. This methodology allows systems to be defined in terms of logically connected function blocks that run on different processing resources. Complete applications, can be built from network of function blocks, formed by interconnecting their inputs and outputs. In a subsequent step the assignment of the application's components into physical resources such as field devices should take place. As far as, the field devices are interconnected with the same fieldbus, the distribution of the IPMCS application is a simple task. However, things are going to become more complicated, when the application's function blocks must be distributed among devices assigned to different fieldbuses of the same or different type. This is a common situation in the industry, where a lot of proprietary fieldbuses have been installed in different time periods under different circumstances. The many different types of commercial fieldbuses resulted in a wide diversity of devices and tools to support the development of IPMCSs.

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) (Lewis 2000). New technologies in Software Engineering as well as modern CASE tools that assist in improving the efficiency of software development process must be considered for the development of distributed IPMCSs.

OT is being recognized as perhaps the best-known way to revolutionize and improve dramatically the efficiency of 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. 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.

All the above guided us, to adopt the Unified Modeling Language (UML) and to examine its applicability in the development process of distributed IPMCSs. We have proposed a new UML based notation to be used instead of the function block one (Thramboulidis, 2001).

In the context of this paper, we consider the development of distributed IPMCS applications over heterogeneous fieldbus segments and we present a framework-based approach. We adopted the Object-Oriented approach to exploit the many benefits of OT as they were encountered in our many projects where we applied it (Thramboulidis, 1997a,b). 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. The proposed Object-Oriented framework assists process and system engineers in the development and configuration of distributed IPMCSs. The whole engineering process should 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 2, we briefly refer to the design process of IPMCS applications using the function block construct as the primary building block. We also present our UML based function block compliant notation for the design of distributed IPMCS applications. Section 3, describes the proposed framework, which address both the IPMCS engineering process, as well as the IPMCS operational phase. Finally, the last section is used to conclude the paper.

2. THE DESIGN OF IPMCS APPLICATIONS

2.1 The Function Block Approach

The function block construct was first introduced by the IEC 1131 standard on programming languages for programmable logic controllers. Evolving international standards IEC 61499 and IEC 61804, ameliorate the function block construct, extending it to the new requirements of distributed control systems. The function block, as is redefined by the IEC 61499, is 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 IEC 61499 further defines a general model and methodology for describing IPMCS applications in a form that is independent from a specific implementation. An application consists of one or more function block *instances,* interconnected through their *event* and *data inputs/outputs,* as is shown in figure 1. Process inputs and outputs define the interface of the application with the controlled industrial process.

The next phase of the development process, after the construction of the function block diagram, results in mapping the functionality captured by the IPMCS application into physical resources such as field devices. According to the IEC61499, d*evices* may communicate with each other over one or more communication links, and may interface to controlled machines and processes. The IPMCS *applications* may be distributed among one or more devices, defining thus the general concept of the distributed IPMCS system.

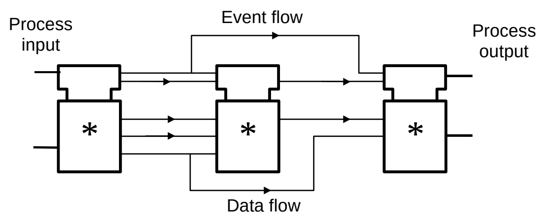

Figure 1. Part of a function block model of an IPMCS application.

2.2 Our UML based Approach

Evolving standards attempt to address the interoperability issue by use of the function block construct. The function block, as is very well pointed out in (Lewis 2000), shares many of the well defined, and already widely acknowledged benefits of the object concepts introduced by the Object Technology. Objects are stable; they reduce the complexity of the system and are highly reusable. The function block shares these attributes and constitutes a well-established concept for defining robust, re-usable IPMCS software components. However the whole approach is procedural like. New advances in Software Engineering, like Object Technology, and component-based development must be seriously considered so as to exploit the maximum benefits introduced by them. In our attempt to exploit the many benefits of the Object Technology we were guided to use the Unified Modeling Language and examine its applicability in the IPMCS area. UML is a general-purpose visual modeling language that is used to specify, visualize and document the artifacts of software system (UML 2000). It is becoming the de-facto standard and is currently adopted by the majority of the design and development tools.

Modeling the function block related concepts

In a first step, we used the UML language to construct a metamodel to capture the function block related concepts. This metamodel clearly represents the concepts and their associations and greatly helped us to manage the complexity of distributed IPMCS applications and to capture the key abstractions required for the design of an Engineering Support System (ESS). Figure 2, shows part of the metamodel we constructed using UML, to clarify the function block related concepts. We have defined the Industrial Process Terminator (IPT) construct, to represent, in the function block diagram, the sources and sinks of the application's events and data. The *connection* abstract-class represents either the connection between two function blocks or the one between a function block and an IPT. In any case, it is a data or an event connection. The aggregation of function blocks, connections and IPTs constitute the IPMCS application. The defined metamodel, constitutes a formal representation of the above concepts, and captures the key abstractions required for the design of an ESS. To further facilitate in the development of an ESS, we have defined two layers of abstraction, the application and the system layer. These two layers provide the necessary mappings between their key abstractions, that the ESS must support.

Towards a UML based notation

In a following step we examined the use of UML in both layers of abstractions. In order to tailor the UML 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. Although UML currently provides only lightweight extension mechanisms, it is expected that heavyweight ones will also be supported in the near future (Kobryn, 1999).

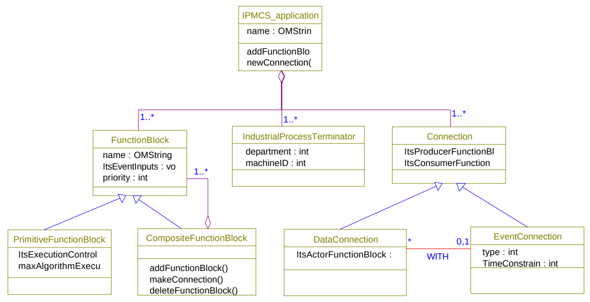

Figure 2. Part of our IPMCS application's metamodel.

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. We defined the following stereotypes as the main building blocks of the IPMCS model:

the FunctionBlock-stereotype (FB-stereotype),
the Data-Dependency-stereotype, and
the Control-Dependency-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. To store the additional properties of the introduced stereotypes, which are not supported by the corresponding base element, we use tagged values. A tagged value of a 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 model elements, the implementation-dependent information, that is needed by code generators and other add-ins such as project planners and report generators.

The Data-Dependency (DD) stereotype is defined as a special kind of association between FB-stereotypes and, FB-stereotypes and Industrial Process Terminators (IPTs). There is a DD-stereotype for each data input, as defined by IEC 61499, of the corresponding function block. 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 so as 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 ESS uses this information to produce the appropriate set or get methods that are used to implement the DD.

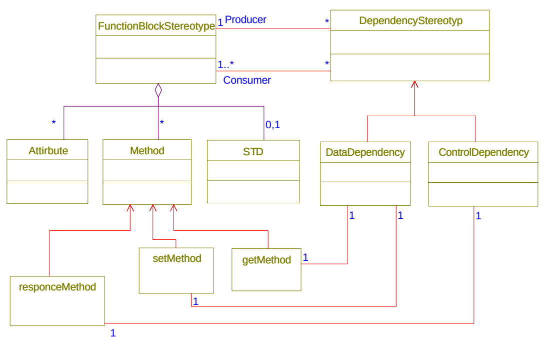

Figure 3. Part of the FB-stereotype's UML model.

The Control-Dependency (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, as defined by IEC 61499, of the corresponding function block. For each CD, there is a corresponding method called response method. This method defines the behavior of the consumer FB to the event captured by the CD. To make a formal representation of the above constructs the UML notation was used. Figure 3 shows part of the derived UML model that clearly represents the above concepts.

We have also considered an alternative for the formal representation of the above constructs. We adopted the meta-model architectural pattern, a well-known infrastructure for defining the precise semantics required by complex software models (Thramboulidis, 2001). We defined a three layer architecture to satisfy the need for IPMCSs models to be reliably stored, shared, manipulated and exchanged across tools.

3. THE PROPOSED FRAMEWORK

Key requirements for distributed IPMCS applications

After the construction of the function block diagram, the mapping of the functionality captured by the IPMCS application into physical resources such as field devices must take place. The IPMCS applications may be distributed among one or more devices. As far as, the field devices are interconnected with the same fieldbus, the distribution of the IPMCS application is a simple task. Data and event connections are mapped into the specific's fieldbus communication mechanisms and real time constraints are satisfied by the fieldbus nature. However, things are going to become more complicated, when the application's function blocks must be distributed among devices assigned to different fieldbus segments of the same or different type. This is a common situation in the industry, where a lot of proprietary fieldbuses have been installed in different time periods under different circumstances. If, for example, an enterprise wants to transfer data between two buildings, with each one having its own fieldbus, it has to interconnect the two heterogeneous fieldbuses. The obvious solution to the above problem is to use interworking units to interconnect each fieldbus segment with the enterprise intranet. Although this solution, address the requirements for the integrated monitoring in corporate level and the development of fieldbus independent engineering support systems, it does not satisfy the requirement for real-time interconnection between fieldbus segments.

Another requirement that constitutes an important issue for the success of a framework is its ability to support the integration of legacy systems and enterprise applications. As legacy systems, we consider the many different fieldbus types already installed in the industry, and as legacy applications, the many control, monitoring and even decision-making applications, which act as SCADA clients systems. The framework must be defined in such a way so as to facilitate the integration of commercial SCADA clients as well as existing fieldbuses.

The framework

We defined an OO framework, to meet the above requirements and the ones imposed by our UML-based approach for the development distributed IPMCS applications over heterogeneous fieldbus segments. 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. The proposed framework would assists process and system engineers in the development and configuration of distributed IPMCSs. The proposed framework address:

a) the IPMCS engineering process, which utilizes the engineering workspace-to-fieldbus channel and
b) the IPMCS operational phase, which utilizes the SCADA client-to-fieldbus channel and the fieldbus-to-fieldbus channel.

Figure 4 shows a high level view of the proposed IPMCS framework. For the SCADA client-to-fieldbus channel, we adopted Internet although its role as a channel over which SCADA applications are built is a fairly recent phenomenon. However 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 engineering workspace-to-fieldbus channel. However, Internet in its current status is not possible to be considered for the fieldbus-to-fieldbus channel, where alternatives such as Fast switched Ethernet, Gigabit Ethernet and ATM may be successfully adopted.

The proposed framework is a set of collaborating classes that embody an abstract design capable to provide solutions for the family of IPMCS's related problems. It will attempt to increase reusability in both architecture and functionality by addressing issues such as interoperability and integrated development of distributed IPMCSs. The main components of our framework are the interworking unit and the virtual fieldbus. The interworking unit will be the infrastructure for the development of the interoperability mechanism, while the virtual fieldbus would provide the APIs required by the different communication channels.

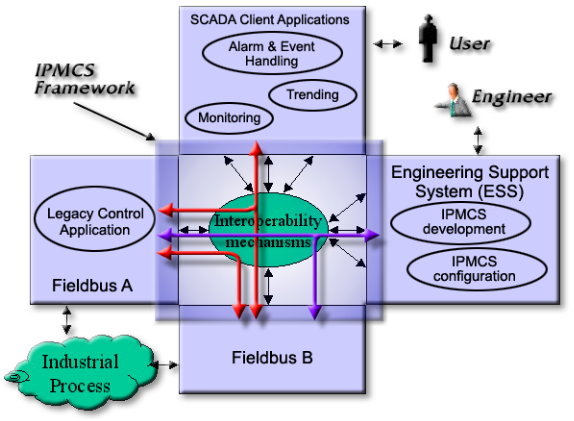

Figure 4. A high-level view of the IPMCS framework.

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 de-facto market standard OLE for Process Control (OPC) was adopted to support the majority of commercial SCADA client systems based on it. Further, the interworking unit was structured in such a way so as to facilitate the integration of existing fieldbuses. CORBA and Java are two already mature software technologies that may be used to achieve the high level of flexibility and reliability required by the IPMCS industry. They have both confronted the interoperability problem a long time before and they have been currently enhanced with features to meet requirements imposed by real-time systems.

With the explosive growth of Internet, emerging standards such as XML make transmitting data over the Web inexpensive and efficient (XML 2000). 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.

The interworking unit

To satisfy the requirement for real-time interconnection between fieldbus segments, we were guided to adopt the network topology shown in figure 5. Our approach to use interworking units between each fieldbus and the enterprise intranet, leaves unchanged the already defined process of each fieldbus and only requires the extra effort for the interconnection of function blocks assigned to different fieldbus segments. This would result, in the least effort that the enterprise should spend over the configuration of the new system.

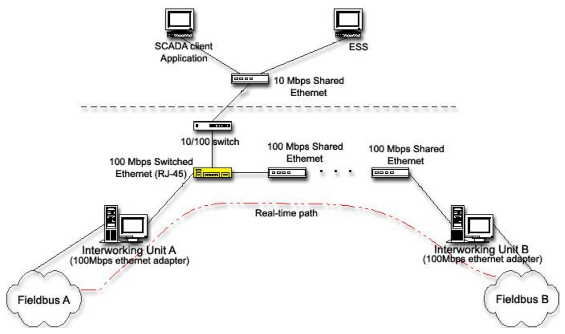

Figure 5. The proposed network topology for the real-time interconnection of fieldbus segments.

The communication subsystem must provide the quality of service required to meet the timing requirements. ATM is one of the successfully used technologies for the interconnection of fieldbuses (Kunert, 1997) (Cseh, 1999). However, to satisfy the key requirement of the simplicity of the communication system as well as the low cost of the equipment, switched Fast-Ethernet was selected.

Virtual fieldbus specification

To provide interoperability in the fieldbus level a virtual fieldbus (VF) must be defined. This VF will allow:

- the interconnection of fieldbus device islands of the various commercial fieldbuses,
- the similes interface to the enterprise intranet and
- the development of fieldbus independent engineering support systems.

a) *the interconnection of fieldbus device islands*

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. Further, the interworking unit was structured in such a way so as to facilitate the integration of existing fieldbuses.

b) *the similes interface to the enterprise intranet*

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. This way, we can interface with no extra cost with the majority of commercial SCADA systems based on it.

c) the development of fieldbus independent Engineering Support Systems

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 (Christian Diedrich, 1998), 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. A lot of virtual fieldbus operations are derived from this model. The whole set of operations that must be supported by the VF's API are extracted by the functions provided by the ESS. The main functions provided by the engineering tool enable the engineer to:

- insert industrial process parameters into the working space, so as to properly define the interaction of the control system with the plant,
- insert field devices to the engineering workspace. The engineer connects the field devices to the proper fieldbus and makes use of device's specifications to create device representatives (device proxies) into the working space,
- design 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,

- design the system. The engineer physically distributes the software building blocks to the available field devices and fieldbuses through their proxies. Partitioning, assignment and scheduling are among his/her most important tasks,
- test the application (optional),
- download the application,
- start and monitor the system (optional).

During the last years a great increase in the number and complexity of commercial CASE tools is encountered. 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 ESS. We adopted an open source CASE tool to form the basis for the development. The use case driven approach has been used for the requirements specification of the proposed tool and 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.

4. 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.

Our primary objective in defining the proposed framework 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 believe that the whole engineering process would be improved in terms of reliability, development time and degree of automation compared with the one proposed by the IEC 61499.

We are currently working on the development of the interworking unit to support the interaction through the selected backbone network, and a prototype of the Engineering Support System.

ACKNOWLEDGEMENTS

This work has been funded in part by the Greek General Secretariat for Research and Technology in the context of the PENED 99 ED 469 project.

REFERENCES

IEC61499, 2000. IEC Technical Committee TC65/WG6, “IEC61499 Industrial-Process Measurement and Control – Specification”, IEC Draft 2000

IEC1804, 1999. IEC Sub Committee No. 65C: DIGITAL COMMUNICATIONS, WG 7: FUNCTION BLOCKS for PROCESS CONTROL , “IEC1804 General Requirements”, IEC Draft 1999

Lewis R.W ,2000. “Modeling Distributed Control Systems Using IEC61499 function block s”, Technical Articles, http://www.searcheng.co.uk/selection/control/tech.htm .

Thramboulidis,K. 2001. “Using UML for the development of distributed IPMCSs”, IEEE Conference on Control Applications.

Thramboulidis,K. et all, 1997a. "REDOM : An OO Language to Define and On-line Manipulate Regulations" Software – Practice & Experience, vol. 27,Oct. 1997.

Thramboulidis,K. et all, 1997b. "Rule Handling in the day-to-day Resource Management problem: an Object-Oriented approach." Information and Software Technology, p. 185-193, v. 39, n. 3, 1997.

UML 2000. “OMG Unified Modeling Language Specification”, Version 1.3, First Edition, Object Management Group Inc. March 2000.

Kobryn, Cris, 1999. “UML 2001: A standardization odyssey”, Communications of the ACM, October 1999, vol.42, No 10.

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

Kunert, O. 1997. “ Interconnecting fieldbuses through ATM”, IEEE international workshop on factory communication systems, 1997.

Cseh, C. et, all, 1999. M.Jansen, J.Jasperneite, “ATM networks for factory communication”, 7th IEEE International Conference on Emerging Technologies and Factory Automation.