2008 IEEE Int Symposium on Computer-Aided Control System Design
Part of 2008 IEEE Multi-conference on Systems and Control
San Antonio, Texas, USA, September 3-5, 2008

Model-Driven design of Industrial Control Systems

M. Marcos, *Member, IEEE,* E. Estevez

Abstract—**Industrial Control Systems are used in most of the industrial sectors to achieve production improvement, process optimization and time and cost reduction. Integration, reuse, flexibility and optimization are demanded to adapt to a rapidly changing and competitive market. There is also a growing requirement that all software tools that support the different phases of the development process (design, configuration, management) can be integrated as well. Thus, a consolidation of modeling methodologies for achieving this goal is needed. This paper proposes a Model-driven approach based on different views of the application (control engineering, electric engineering and software engineering views) for designing industrial control systems. The approach is based on the definition of a description language that includes the description of the three domain views and their relationships. In particular, XML schema technology is used to define the architectural style and schematron technology to implement the composition rules.**

I. Introduction

Industrial control systems are specific computer systems not only because of the special characteristics of the environment they control (industrial processes) but also because they use specific equipment, such as Programmable Logic Controllers (PLCs), industrial communication systems (fieldbus), and dedicated I/O devices. On the other hand, as a computer system, its design deals with the definition of the software architecture. Traditionally, each control equipment vendor offered a development system based on proprietary software architectures and programming languages. In the last years, a great effort is being done by international organizations for promoting the use of standards for this applications field. PLCOpen [1] was founded in 1992 and it is a vendor- and product-independent worldwide association whose mission is to be the leading association resolving topics related to control programming to support the use of international standards in this field. The International Electrotechnical Commission (IEC) promotes open systems to be used in the industrial control field. The IEC 61131-3 standard [2], [3] proposes a software model and programming languages for Industrial Process Measurement and Control Systems (IPMCS). Most of the PLC manufacturers are aiming at becoming IEC 61131-3 standard compliant. Furthermore, as fast as industry reaches a greater maturity level and applications become more complex, a consolidation of methodologies, which allow system description and definition before its construction, becomes more necessary.

As fast as industry reaches a greater maturity level and applications become more complex, a consolidation of the modeling methodologies becomes necessary. Therefore, modeling languages, that allow system description and definition before construction, must be used. In this sense, Model-Driven Engineering (MDE) technologies offer a promising approach to address the complexity of platforms as well as to express domain concepts effectively. Model-Driven Development (MDD) is an emerging paradigm which solves a number of problems associated with the composition and integration of large-scale systems while, at the same time, allows incorporating the advances in software development, such as component-based middleware. The MDD approach relies on the use of models to represent the system elements of the domain and their relationships. In MDD, models are used in most system development activities, i.e., models serve as input and output at all stages of system development until the final system is itself generated [4], [5], [6]. Component-based Software Engineering (CBSE) is a relatively new software engineering approach that has been proven to be very powerful in the design of software intensive systems [7], [8], [9], [10].

This paper presents a Component Based Approach as applied to Industrial Control Systems, using the basic concepts of CBSE for defining the role of component and connector that allows defining an industrial control system. Section 2 analyses other approaches for defining IPMCS. In Section 3 a multidisciplinary domain modeling is proposed that identifies the semantic of component and connector in each domain view. In section 4, XML technologies are investigated to be used not only for defining the Domain Description Languages but also to implement the composition rules that assure a correct description of an application. Section 5 presents the modeling of a typical industrial application: a Heat Treatment Line. Finally, the conclusions section points out other XML technologies that could be used for developing an Integrated Development Environment that based on the proposed modeling, integrates the tools involved in the design of the target applications.

II. Related work

Some attempts have been done by different authors towards the use of well-known modeling languages for

Manuscript received January 15, 2008. This work was financed in part by the MCYT&FEDER under project MEC 2006-4003 and by UPV/EHU under project DIPE 06-16.

M. M., and E. E. Authors are with the Department of Ingeniería de Sistemas y Automática, University of the Basque Country, Spain (e-mail: marga.marcos@ehu.es, elisabet.estevez@ehu.es,).

designing IPMCS. [11], [12] propose the use of the Unified Modeling Language (UML) [13], [14] for specifying components of control systems following the IEC 61131-3 standard. Both approaches apply Object-Oriented technology to model industrial automation applications. There are other works that focus on the design of applications using the IEC 61131 standard. [15] proposes a methodology for the analysis and modeling of discrete event systems applied to the development of the control logic. This methodology is offered jointly with a tool that generates an IEC 61131-3 program expressing the execution control of the application.

Related to the distributed IPMCS, the evolving IEC 61499 standard [16], [17] is being discussed as a successor of the IEC 61131 standard. Currently, there is a great interest on analyzing the influence of the new standard in industry, as well as its convergence with the IEC 61131-3 standard. Several groups focus their research in this topic. For instance, [18] discusses the role of the IEC 61499 in factory automation and it analyses the current on going research by several groups. [19], [20], [21] describe a framework called CORFU (Common Object-oriented Real-time Framework for the Unified development of distributed IPMCS application) that defines an Industrial Process-Control Protocol for the development, distribution, and operation of IPMCS applications based on IEC 61499 Function Blocks. The design is object oriented and uses UML to define a 4-layered architecture for designing such type of systems [21].

Nowadays most of the PLC manufacturers offer programming tools that follow the IEC 61131-3 standard, and, in most cases, they provide a proprietary protocol for distributed applications involving their own PLCs. What the IEC 61131-3 standard does not solve is the interoperability among PLC programming tools that would allow true software reuse. In this context, the Technical committee 6 (TC6) of PLCopen organization [1] has selected the eXtensible Markup Language (XML) [22],[23], to define a common language for transferring the graphical information that appears in the screen among IEC 61131-3 compliant tools. The TC6 has defined a global XML schema [24], for defining automation projects with the elements defined in the IEC 61131-3 software model. As the aim is to communicate programming tools, it is not necessary to check for consistency (the transferred code is supposed to be tested in the both end tools).In the same way, CAEX [25] appears to exchange data between process engineering tools and process control engineering tools. Thus, the proposed schema represents the system topology but not checks the information coherency and consistency.

The goal of this paper is to define a component based modeling of industrial control systems and to propose a generic and extensible Markup Language (ML) that specifies not only the elements but also the composition rules. The domain modeling approach can be used as a means to integrate other tools than programming, such us visualization, design or graphical modeling tools [26], [27], [28], [29] [30], [31]. Other XML technologies can be used to achieve tool integration [32], [33], [34], [36], [37]. In [38] an Integration Development Environment is presented that uses the Markup Language proposed in this paper to integrate a set of Commercial-Off-The-Shelf (COTS) tools for supporting the development cycle of industrial control applications.

III. Component based modeling for Industrial Control Systems

The design of distributed industrial control applications involves different domain fields. Each domain defines a different view of the same system: the ***control engineering domain*** defines the functional design that contains the control strategies (*"what"* the control system has to do in order to meet the functional and non functional requirements).

The ***electronic- electric engineering*** domain deals with the selection of the equipment needed to start-up the control system. The hardware architecture commonly consists of a set of network nodes (nodes) connected through a set of network segments (buses). A Node element is connected to a network segment through a communication board. There can be two types of nodes: those that contain processing resources (e.g. PCs, PLCs, OpenPLC) and those that are only used as input/output devices (e.g. PROFIBUS_DP slaves). The former contains processing resources (CPUs) and other type of boards (memory cards, IO cards,).

Finally, the ***software engineering domain*** deals with the software architecture which implements the functional specification. It defines the software architecture of the application following the IEC 61131 standard software model. The standard provides different elements that allow defining the software architecture [2]. A *Configuration* represents a PLC or Open PLC, the *Resource* provides support for program execution (a CPU or a Virtual machine), a *Task* allows the designer to define the execution rates of different parts of the program. Finally, *POUs* (*Program Organisation Units*) are the elements that provide software reuse. Three types can be distinguished: the *Program*, the *Function Block* (*FB*) and the *Function*. They define and/or use Variables, characterized by their visibility, type and value.

These two latter views of the system define the implementation issues ("how" to develop the functional specification).

A Component based Approach to describe this type of applications consists of identifying the components and connectors that can be used to define the three domain views of an application, as well as the constraints that each domain view must fulfill in order to connect components through connectors. Furthermore, given that all of them represent the same application, they are not completely independent and

to assure consistency, components and connectors belonging to the different views must meet some relationships. Next sub-sections identify the components and connectors of each domain.

A. Control Engineering Domain

The functional specification of an industrial control system contains the control strategies to be implemented that assure that the system under design fulfilled the functional and non-functional requirements. In order to describe a functional specification in a component based approach, the following elements are needed: component, connector, channel and port.

Fig. 1 shows the elements and the architectural style of the control engineering view. These elements allow describing a hierarchical specification having enough detail so as to give freedom to build upon it different graphical representations, independently of the number of components and connectors present in the application.

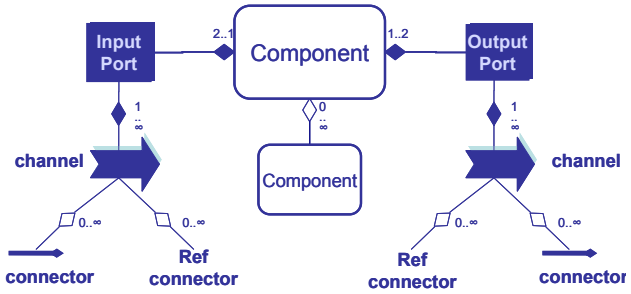

Fig. 1: Elements for the models of the functional specification

A ***functional Component*** is a black box grouping the control application functionality or part of it. It may contain a set of components of an immediate lower level. Thus, a hierarchical specification is achieved where each component is characterized by the level of the hierarchy it belongs to. Each component can be characterized by the requirements in which it is involved. This makes easy to implement requirement traceability.

A ***functional Connector*** represents the interaction between components, i.e. transmission of information. Two types of connectors can be distinguished: a Field Connector contains information coming from / going to the process (sensors and actuators), Human Machine Interface (HMI) and/or operator desk, i.e. field signals. An Internal Connector represents the communication between functional components of the same hierarchical level.

Within typical industrial control applications, the number of field signals that the control system processes can vary from dozens to hundreds. In order to allow having a compact representation of a certain level of the hierarchy, two more elements are defined: channels and ports. ***Channels*** allow grouping a set of connectors that come from / go to the same component. Every channel contains at least one connector. ***Ports*** represent the point in which channels are related to components. A component contains at least one input and one output port. A port contains at least one channel. In order to distinguish inherited connectors (top-down or bottom-up) from connectors that represent communication between components at the same level, two types of ports are defined: vertical and horizontal. A ***Vertical Port*** is used by channels that contain inherited connectors. A *Horizontal Port* is used for communicating components at the same level.

B. Electronic –Electric Engineering Domain

The hardware architecture is described in two levels. The first level describes the nodes interconnected by bus segments. The second level describes each node by means of the modules and boards that compose it.

Therefore, the ***hardware Components*** are bus segments, nodes, modules and boards, except the communication board as it represents a type of connector. The common characteristics of all components are the manufacturer, serial number and firmware version. Other are manufacturer dependent and thus, specific of every equipment. The components that describe the bus segment are characterized by the speed and the protocol of the OSI Application layer.

Related to ***hardware Connectors***, there is one type for each description level. Every communication board that connects the node to a bus segment represents a *Segment-connector*. The Message-connector is used at the second level diagram, and it represents the data contained in I/O cards. In particular, a message-connector is associated to each byte in a digital card, or to each analogue channel. In both cases, the connectors are characterized by their relative address in the card and its size.

Fig. 2 illustrates how hardware components are defined by extending the basic characteristics with manufacturer information.

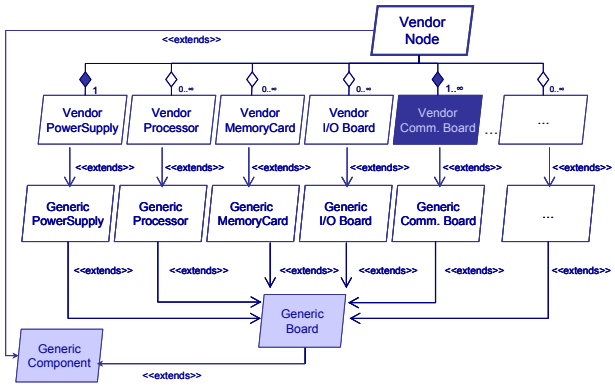

Fig. 2: Definition of hardware components by means of extension mechanism

C. Software Engineering Domain

Software Components correspond to the elements provided by the IEC 61131-3 standard. *Configuration*, *Resource* and *Task* organize the software architecture and POUs (*Program*, *FB* and *function*) encapsulate code. A Configuration represents the complete software architecture

of a PLC. It contains at least one Resource. Each Resource contains the executable code structured in Programs. The timed execution is achieved by associating Programs and Function Blocks (*FBs*) to tasks. In this sense, all the programs and FBs associated to the same task execute at the same period and priority.

Software **connectors** correspond to the IEC 61131-3 variables and they represent the communication between different components. In fact, their visibility (VAR_ACCESS, VAR_GLOBAL) identify the components that communicate.

Fig. 3 illustrates the elements and the architectural style of the software view.

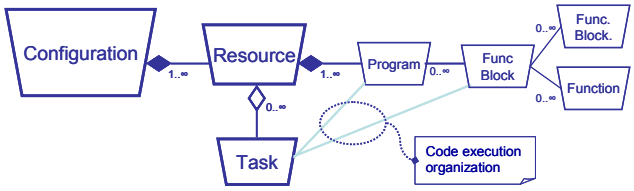

Fig. 3: Elements and architectural style of the software architecture

D. Relationships among domain views

The three domains commented above are not independent. In order to define a consistent application, their components and connectors must be related among them. A functional component is related to the hardware component in which it is implemented. Field Connectors in the functional domain are related to message-connectors of the hardware architecture.

A functional component is related to the software component that implements it. A functional connector can be related to a variable in the software architecture.

Finally, a software component can be related to the PLC in which it has to be downloaded. On the other hand, each message-connector of the hardware architecture corresponds to a global variable in the software architecture. Finally, if the control system consists of more than one PLC, the communication between them requires a relationship between every communication board (segment-connector) and an IEC 61131-3 access variable that contains the exchangeable information.

IV. Industrial Control System Description Language

The generic language includes the three domain languages as well as the relationships among the three views. The selected platform for implementing the languages has to provide mechanisms not only for defining the elements and the architectural style of each view, but also for implementing the composition rules. Finally, if each part of the model can be generated / consumed by different software tools, the platform should provide means for extracting part of the information contained within a model description. XML technologies satisfy all these requirements. There are different XML standards that can be used not only for describing Industrial Control Systems, but also for performing model coherency and consistency checks as well as for filtering, processing and formatting part of the information contained in the model.

XML schema [24] standard allows defining a new Markup Language (ML) using a vocabulary of components, connectors and constraints. These elements can be extended through properties and attributes and they are organized by relationship types (a sequence of elements, a choice among elements, at least one element, etc.). Each domain view ML consists of the definition of its architectural style (the grammar) as well as a set of composition rules for assuring model consistency. These latter apply to domain view components, connectors and to the relationships among domain views.

The characteristics of each element are implemented as XML attributes. When necessary, new XML simple types have been defined for defining the range of values and the presentation pattern. The architectural style is defined by a root element that represents the domain view and the relationships among the XML elements of the domain view. The mechanisms used to relate the elements of the language are XML schema *sequence* or *choice* elements that can have *multiplicity*.

The XML schema standard can be complemented with XML schematron rules [35] in order to assure the fulfillment of the composition rules. These have been formulated making use of Xpath sentences [37] following the abstract syntax described in [38]: $\forall x \in XPath$ *(condition to check)*. Where $\forall x \in XPath$ is the same as $x = \mathbf{S}\,[[p]]_y$, being x the set of selected nodes of p pattern, having y as a context node. The condition to check has the following structure:

$condition|condition \wedge condition|condition \vee condition|\neg condition|condition \rightarrow condition$, where the Basic condition is the same as $\mathbf{Q}[[q]]_z$.

The composition rule is constituted by two parts: the first one consists of allocating the context node. The second part indicates the condition to be checked. Note that these rules return a Boolean value. A TRUE return value means that the context node follows the composition rule. Otherwise, an error message is returned.

The complete markup languages can be found in http://www.disa.bi.ehu.es/gcis/projects/merconidi/icsML. The generic markup language imports the three domain languages adding the relationships among them.

V. An Industrial Case Study: a Heat Treatment Line

This section illustrates the proposed modeling of distributed IPMCS as applied to an industrial case study: the distributed control system of a Heat Treatment Line. These lines are commonly used for manufacturing metallic elements in many industries and its main goal is to provide the required quality characteristics for the final product. A typical line is composed by: Furnaces, transport systems to charge and discharge the material, cooling tanks, washing

and degrease machines and presses.

Fig. 4 illustrates a typical Heat Treatment Line (HTL) that is composed by the following sub-systems: a Load System, an Austenizing Furnace, a Tempering Tank, a Washing Tank and an Annealing Furnace.

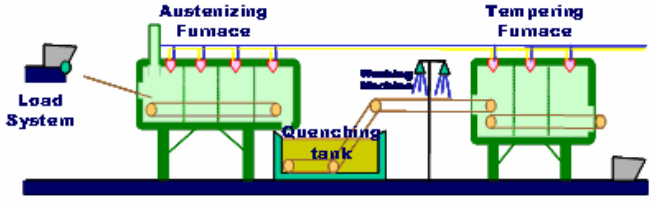

Fig. 4: General overview of a Heat Treatment Line

Let us apply the proposed modeling to two representative sub-systems of the complete line: the Austenizing Furnace and the Load System.

The design of the control system functionality for the complete line involves four hierarchical levels:

- **Level 0** represents the plant.
- **Level 1** is composed by components corresponding to each independent subsystem of the plant.
- **Level 2** defines the set of components within every component of level 1. For instance, the Austenizing Furnace, a level 1 component, includes 6 level 2 components: the Gas Train Control, the Burner Combustion Control, the Zone Fan Control, the Combustion Fan Control, the Temperature Regulation and the Movements Control.
- **Level 3** is composed by elementary functional components. For instance, the level 2 component Zone Control contains three elementary blocks: the control of the temperature, the burners and the fan.

Fig. 5 illustrates the HTL functionality expressed as a XML file that follows the control engineering ML. It shows the level 0 component representing the plant, its input port and the group of channels it contains. This figure also illustrates the hierarchical specification: the level 0 component contains two components of level 1 which contains level 2 components.

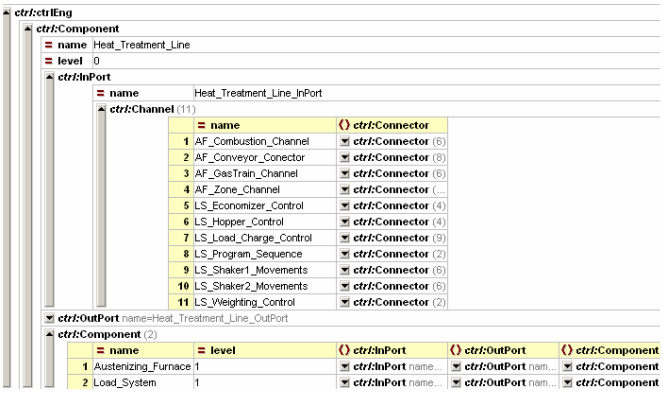

Fig. 5: The functionality of the HTL in XML

The hardware architecture defines two controllers (one is a OMRON Open Network Controller and the other a S7300 PLC from Siemens manufacturer), being both the master of a PROFIBUS-DP segment. Fig. 6 illustrates part of the XML model instance that corresponds to the hardware architecture. In particular, it illustrates the characteristics of the processing node S7300 that contains a set of boards. Fig. 7 shows the software architecture for this application. As it can be seen, the project automation for each PLC is described, following the architectural style commented above.

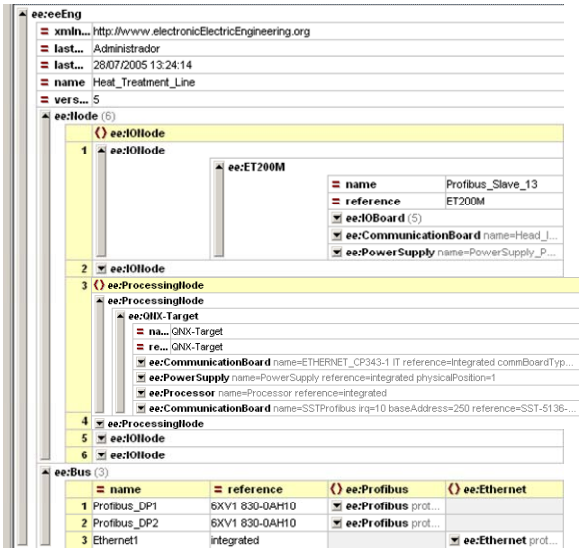

Fig. 6: Hardware architecture of the HTL

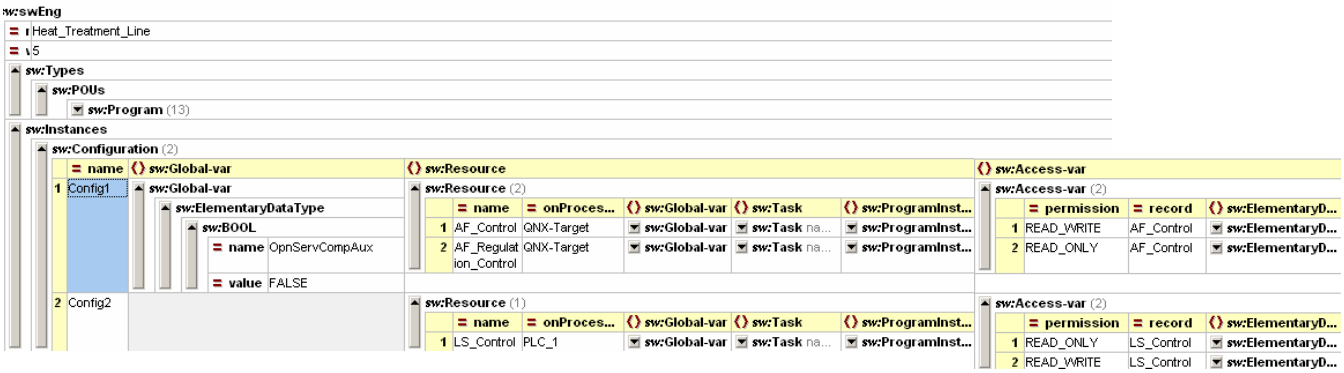

Fig. 7: Software architecture of the HTL

VI. CONCLUSIONS

This work has investigated the use of CBSE as applied to Industrial Control Systems. Three different views have been identified as the domains involved in the design of such type of systems. The concept of component and connector drawn in CBSE has been used in order to identify its role in each domain, as well as the relationship among the components and connectors of the different views.

The platform for implementing the proposed modeling language has been selected to meet other requirements than expressive power. Besides describing an application in terms of a hierarchical functionality and the hardware and software architectures, the platform should allow analyzing the correctness of the description (including the relationship among views), offering mechanisms for extracting the information corresponding to a simple view (filtering), mechanisms for generating and/or modifying a view. XML technologies meet these requirements through XML schema, XML schematron and XML stylesheet technologies. Furthermore, they also offer graphical languages (such as Scalar Vector Graphics) that can be used to create an abstract layer offering to the user a graphical tool for specifying the application model.

An implementation using XML technologies for the three description languages have been proposed. These XML-based descriptions are going to be used for supporting the development cycle of the application. The XML stylesheet technology jointly with DOM can be used to integrate the set of COTS tools involved in the application design. In fact, these tools will generate/consume the different domain models. In summary, the use of open software technologies has been proven to be directly applicable in the automation and control field.

REFERENCES

[1] PLCOpen international Organization, available at: http://www.plcopen.org/

[2] Lewis, R.W. "Programming Industrial Control Systems using IEC 1131-3" (1998). IEE Control Engineering Series. 1998.

[3] John, K-H and Tiegelkamp, M., "IEC1131-3: Programming Industrial Automation Systems". Springer. 2001.

[4] Developing Applications Using Model-driven Design Environments. Krishnakumar Balasubramanian, Aniruddha Gokhale, Gabor Karsai, Senior, Janos Sztipanovits, and Sandeep Neema.

[5] Miller, J., Mukerji J. Model Driven Architecture (MDA). OMG, ormsc/2001-07-01, Architecture Board ORMSC1, July 2001

[6] Miller J y Mukerji J. MDA Guide Version 1.0.1. Document -- omg/03-06-01 (MDA Guide V1.0.1). June, 2003.

[7] Crnkovic, I., Schmidt, H., Stafford, J., Wallnau, K.. "Automated Component-Based Software Engineering ".Journal of Systems and Software, 1,1. 2005.

[8] Crnkovic, I., Larsson, M., 2002. Building Reliable Component-Based Software Systems. Artech House publisher, ISBN 1-58053-327-2.

[9] Schmidt, H., 2003. Trustworthy components: compositionality and prediction. Journal of Systems and Software 65 (3), 215–225.

[10] Medvidovic, N., Taylor, R.N., 1997. Exploiting architectural style to develop a family of applications. IEE Proc. Software Eng. 144 (5–6), 237–248.

[11] Heverhagen, T., Tracht, R. (2001) "Integrating UML-RealTime and IEC 61131-3 with Function Block Adapters". Proceedings of the IEEE International Symposium on OO RT Distributed Computing.

[12] Jacobson, I., Christerson, M., Jonsson, P., Övergaard, G., (1992) "Object - oriented software engineering". Addison-Wesley.

[13] Rumbaugh, J., Blaha, M., Premerlan, W., Eddy, F.,Lorensen, W., (1996) "Object-Oriented Modelling Design". Prentice Hall.

[14] Gonzalez, V. M., F. Mateos, N. Amos (2003). "MLAV. Object-Oriented Methodology for the Analysis and Modelling of the Control Logic of Discrete Event Systems", SSGRR 2003

[15] International Electrotechical Commision, IEC International Work in Progress, IEC 61499-3, 2004. "Function Blocks for Industrial Process Measurement and Control systems".2004.

[16] Lewis, R.W. (Robert W.), "Modelling control systems using IEC 61499 Applying function blocks to distributed Systems". The Institution of Electrical Engineers ,2001.

[17] K. Thramboulidis (2005)."IEC 61499 In Factory Automation". Int. Conf. on Industrial Electronics, Technology&Automation (CISSE-IETA 2005).December 2005.

[18] Thramboulidis, K. (2004). "Developing a CASE tool for distributed control Application". Int J Adv Manuf Technol (2004). 24: 24-31

[19] CORFU Framework, available at: http://seg.ee.upatras.gr/corfu/dev/index.htm

[20] K. Thramboulidis (2002). "Development of Distributed Industrial Control Applications: The CORFU Framework". In Proc of 4th IEEE International Workshop on Factory Communication Systems, Vasteras, Sweden, August 28-30, 2002.

[21] K. Thramboulidis, C. Tranoris, (2001). "A Function Block Based Approach for Development of Distributed IPMCS Applications". In Proc of the 10th IEEE International Conference on Advanced Robotics (ICAR 2001), August 22-25, 2001, Budapest, Hungary.

[22] XML, available at: http://www.w3.org/XML/

[23] Pruitt et al (1998).Steve Pruitt, Doug Stuart, T.W. Cook. "The merit of XML as an Architecture Description Language Meta-Language". Microelectronics and Computer Technology Corp, October 1998.

[24] XML schema, available at: http://www.w3.org/XML/Schema

[25] RWTH Aachen, Website of CAEX at: http://www.plt.rwth-aachen.de/index.php?id=228&L=1index.php%3Fid%3D141 . 2008.

[26] Bani Younis M.; Frey, G.: "Formalization and Visualization of Non-binary PLC Programs". Proceedings of the 44th IEEE Conference on Decision and Control (CDC 2005) and European Control Conference (ECC 2005) Seville, Spain, pp. 8367-8372, Dec. 2005.

[27] Bani Younis, M.; Frey, G.:"UML-based Approach for the Re-Engineering of PLC Programs". Proceedings of the 32nd Annual Conference of the IEEE Industrial Electronics Society ,Paris, France, pp. 3691-3696, November 2006.

[28] E. Estevez, M. Marcos, N. Iriondo, D. Orive, "Graphical Modelling of PLC-based Industrial Control Applications". Proceedings of 26th American Control Conference, New York, USA, July 2007.

[29] E. Estévez, M. Marcos, I. Sarachaga, D. Orive." A Methodology for Multidisciplinary Modeling of Industrial Control Systems using UML", .Proceedings of The 5th International Conference on Industrial Informatics (INDIN 2007). Viena, Austria, July 2007.

[30] XSL available at: http://www.w3.org/TR/xsl

[31] XSL-FO available at: http://www.w3schools.com/xslfo/default.asp

[32] DOM: http://www.w3.org/DOM/

[33] SAX http://www.saxproject.org/

[34] E. Estévez, M. Marcos, U. Gangoiti, D. Orive (2005)."A Tool Integration Framework for Industrial Distributed Control Systems". In Proc of the 44th IEEE Conference on Decision and Control and European Control Conference, pp. 8373-8378, CDC-ECC, Seville, Spain (2005).

[35] Rick Jelliffe schematron rules, available at: http://www.ascc.net/xml/schematron/

[36] Van der Vlist, E. XML schema, Ed. O'REILLY. 2002.

[37] XML Path Language. Available at: http://www.w3.org/TR/xpath

[38] Philip Wadler, 2000. "A formal semantics of patterns in XSLT". Bell Labs, Lucent Technologies.