Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

A Tool Integration Framework for Industrial Distributed Control Systems

E. Estévez, M. Marcos *Member IEEE*, U. Gangoiti, D. Orive

***Abstract*— This paper presents a tool integration approach for supporting the development cycle of industrial distributed control applications. The core of the approach is a formal model of the application that expresses separately the functionality of the control system from the implementation issues. The modeling language selected is XML (eXtensible Markup Language) and the framework proposed makes use of XML technologies for achieving the integration of the tools involved within the development cycle. Following this approach, a set of tools from different domain of expertise (control engineering, software engineering, configuration, maintenance…) have been integrated into a framework. The toolset has been used for designing and validating the distributed control system of a heat treatment line.**

I. Introduction

NOWADAYS ,most of the industrial sectors use distributed *Industrial Process Measurement and Control Systems* (IPMCS) to achieve the control of their productive systems. Integration, reuse, flexibility and optimization are highly demanded in these types of applications in order to adapt to a rapidly changing and competitive market.

The technological advances in the last years in both hardware equipment (e.g. programmable robots, PLCs, Open PLCs, intelligent embedded controllers, smart sensors and actuators, communication networks, …) and software tools (for modeling, control system design and simulation, code generation, configuration, etc) allow production improvement, process optimization, time and cost reduction.

On the other hand, there is strong movement towards the definition of standards to achieve both standardized and open systems. In this sense, application of standards has also a great force in the fast growth of the control and instrumentation of industrial processes. In this sense, the *International Electrotechnical Commission* (IEC) has published several standards promoting interchangeable and open systems. With respect to the software production, the IEC 61131-3 standard [1] defines a software model and programming languages for IPMCS. Besides that, and corresponding to the industrial networks, the IEC 61158 defines a set of communication protocols for process automation applications. Among them, three of the most used are: PROFIBUS, Foundation Fieldbus and CAN.

Manuscript received March 4, 2005. This work has been supported in part by EU-IST programme under project IST-2001-37269 and by MCYT&FEDER under project DPI 2003-2399.

E. E., M. M., U. G, O. D. are with the Departamento de Ingeniería de Sistemas y Automática from the University of the Basque Country, Spain. (email: elisabet.estevez@ehu.es, marga.marcos@ehu.es, unai.gangoiti@ehu.es, dario.orive@ehu.es).

The current market offers well-known software packages that can be used during certain phases of the design of this type of applications. Usually, they make use of graphical techniques based on different mechanisms that assist the designer to build a model of the system. In the control engineering domain, for instance, the system is represented by a set of transfer functions that represent the individual components of the overall system. In discrete event-driven systems, the model represents the different states that the discrete system can reach, the events that provoke transitions between states and the actions that must be carried out in response to certain events. During the software design phase, it is possible to use commercial *Computer Aided System Engineering* (CASE) tools, based on well-known software development methodologies. During the coding phase, each PLC manufacturer offers a programming environment where the user can define the software of the application as well as specify the hardware of the implementation. Therefore, depending on the design phase, there are graphical tools that allow a model of the system to be obtained. The common tendency of such tools is to offer the user a graphical editor in order to build the system model from a set of simple blocks that can be connected. Besides the graphical specification, most of the tools offer the option of simulating the system behavior from the model.

There is a growing requirement that all these software tools supporting the different phases of the development process (design, configuration, management) can be integrated. Thus, a consolidation of modeling methodologies for achieving this goal is needed.

However, as far as the authors know, there is the lack of a commercial environment that integrates automatically all design phases, from specification to code generation of application.

Some software vendors are adopting a tool integration approach, following the demand from final developers of IPMCS applications. The current market offers well-known multi-disciplinary *Commercial-Off-The-Shelf* (COTS) tools that can be used along the application development cycle. There are some software packages for system analysis and simulation that allow some exchange of information between the tools that cover the design phase. For instance, Extessy [2] integrates the Matlab modeling and simulation tool [3] with ARTiSAN RtS UML tool [4]. HyPneu [5] integrates hydraulic and pneumatic models to Simulink applications; CosiMate [6] integrates Mechatronics and Simulink. Tool integration has also been investigated in the automotive sector integrating ADAMS and Xmath tools [7]. In all cases, integration is achieved for proprietary tools and they collaborate within a closed environment, covering a certain

0-7803-9568-9/05/$20.00 ©2005 IEEE

phase of the design cycle. It is not possible to easily communicate the integrated environment offered by a software vendor with other tools of the user company. The key factor is that there are neither methodologies nor middleware software that supports the integration of all these type of components.

The phases involved in development cycle of these applications are the typical phases involved in a software / hardware engineering project, where the design tools and the equipment belong to the application field: analysis and simulation tools (e.g. the Matlab environment [3], Simulink), PLCs programming tools (e.g. ISaGRAF Enhanced [8], Simatic Administrator [9]), software design tools (e.g. UML [4]), configuration tools, documentation tools, etc. But, as far as authors know, there is a lack of environment that automatically integrates all design phases, making the tools to collaborate among them.

In summary, as fast as industry reaches a greater maturity level, a consolidation of the modeling methodologies becomes necessary. Therefore, modeling languages, that allow system description and definition before their construction, must be used. The modeling methodology should allow to model the system from different viewpoints as well as to involve the entire system development cycle. In this sense, PLCopen [10] 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. For this, PLCopen has several technical and promotional committees (TCs). In particular, TC6 for XML [11] had as original goal to define an open interface to communicate different PLC programming tools. But, from the beginning, TC6 members realized that having XML as a common road, other tools, like simulation and modeling tools, or documentation and version control tools could be integrated.

In this paper, a tool integration approach is presented that is based on the definition of a formal model for IPMCS proposed by authors in previous works [12]. The application model captures all aspects of the system to design in terms of functionality and implementation issues (hardware and software). The technologies selected for implementing the tool integration framework are based on XML [13]. In particular, XML *schema* [14] and *schematron* rules [16] allow implementing model validation, and XML *stylesheet* [15] allow easily transforming information coming from / going to different software tools, achieving tool integration.

The layout of the paper is as follows: section 2 presents the different phases of the development cycle of the distributed control system applications, illustrating the most common tools used in each phase. Section 3 describes briefly the formal modeling for this type of applications as it is the core of the proposed framework. It also explains the steps to follow to integrate a tool within the framework. In section 4 the toolset developed within the European project FLEXICON IST-2001-37269 for designing industrial control applications is presented, illustrating the integration of tools offered in every design phase.

II. DESIGN PHASES FOR DISTRIBUTED CONTROL SYSTEMS

This section briefly describes the main phases of the development cycle for IPMCS applications as well as the heterogeneous tools that are involved during the design. In this sense, the typical software engineering phases are:

Requirement Analysis Phase. Usually the application must meet a set of functional (related to process control, logic control, alarms, etc.) and non-functional requirements (e.g. those related to temporal restrictions of control system reaction).

Analysis Phase. During this phase different type of analysis and simulations are performed using the appropriate engineering tools. Thus, the significant process variables are identified (controlled, manipulated, disturbances), process models are obtained and control strategies are decided in order to meet the functional requirements. The selection of the instrumentation needed is also decided as well as the need of networks depending on the number of I/O signals and the physical distribution of the process. Thus, from this phase, a high level design of the control system emerges, as well as, the needs of distribution.

Design Phase. During the system design phase, the high level hierarchical design is modeled. This corresponds to the functional description of the control system that is independent from the implementation issues. This functional model can be implemented in different hardware architectures and with different software architectures that are defined during the detailed design. For IPMCS, the software architecture follows the software model defined by the IEC 61131-3 standard.

Coding Phase. In this phase, the source code of the programs within the resources of each configuration is programmed. To do it any of the five programming languages provided by the IEC 61131-3 standard can be used. The last part of the coding phase is to generate the automation project for the PLC programming tool.

Testing Phase. Usually, the detailed design of the previous phase is developed incrementally by coding parts of the application, performing unitary integration and validation tests (to assure that the functional and non-functional requirements are met). Thus, coding and testing phases are used iteratively for achieving control system validation.

Maintenance Phase. Complex software systems are in continuous evolution. Thus, they should be maintained and extended according to the client necessities.

III. A TOOL INTEGRATION FRAMEWORK

As it has been highlighted in the previous section, a set of COTS tools and / or proprietary tools are used within each phase of the application development cycle. Usually this set of tools are heterogeneous with respect to the domain of expertise they belong to (control engineering, modeling, simulation, code generation, configuration, etc.), as well as

with respect to the hardware and software platform in which they execute.

This work proposes a tool integration framework that is based on a formal modeling of the overall distributed control system [12] and provides the necessary mechanisms to integrate tools on it. Thus, the framework not only achieves tool integration but also tool collaboration. The modeling language is XML and the use of these technologies allows integrating tools that generate / consume information, assuring the model coherency and consistency after any change.

A. Formal Modeling based on XML Modeling Language

The model is divided in two complementary parts: a modular and hierarchical specification of the control system (functional specification), and the implementation issues (architecture), that are expressed as a model of the hardware components and the software architecture for each of the processing elements. Within this application field, the control system is usually implemented in PLCs or Open PLCs. Thus, the software architecture follows the software model proposed by the IEC 61131-3 standard [17]. The model is completed with the relationship between the different parts, as the components of the functional specification are mapped to specific processing elements (hardware components) and each processing element has specific software architecture.

Fig. 1 illustrates the grammar defined expressed as a XML schema (in order to make it more readable, *WSDL design view* offered by XMLSPY™ tool is used). An application model is composed by three elements: the functionality, the implementation (hardware and software architectures) and the relationship between them.

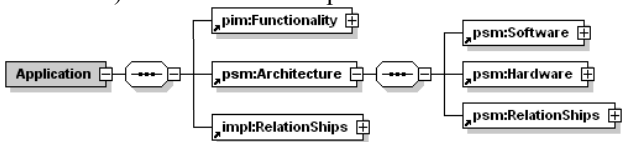

Fig. 1. Model general overview

The *Functionality* XML element defines how to model the hierarchical specification of the control system, independent from the technologies used for its implementation. It defines a generic hierarchical specification based on components and connections.

The software architecture (*Software* element) models separately the user defined types (POUs or derived data types) from the application software model (following the IEC 61131-3 standard).

Fig. 2 illustrates the associated XML schema.

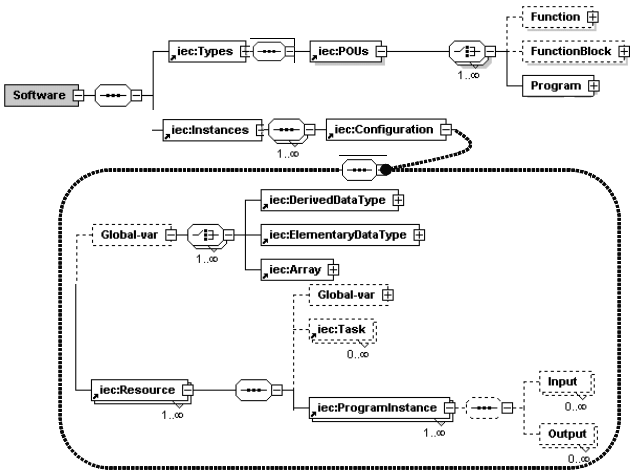

Fig. 2. IEC 61131-3 SW model for application SW Architecture

Usually these types of applications are implemented using specific control equipment. Most of the installed applications use proprietary controllers, and thus, the characteristics and configuration of the hardware equipment is manufacturer-dependent. When distribution is necessary, the control system is extended with network segments and IO nodes. In this sense, the hardware architecture of the control systems is usually constituted by the same conceptual items but the programming, configuration and operation may differ depending on the supplier. Thus the hardware architecture (*Hardware*) is modeled using sets of pre-defined HW templates for controller nodes, I/O nodes and network segments as Fig. 3 illustrates.

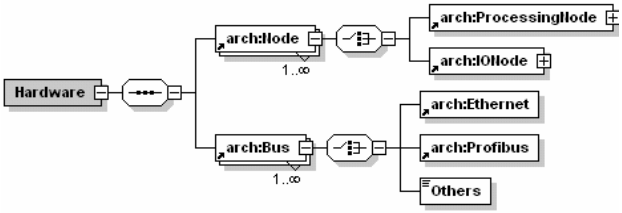

Fig. 3. Hardware architecture

For instance a *ProcessingNode* S7300 from Siemens manufacturer is illustrated in

Fig. 4.

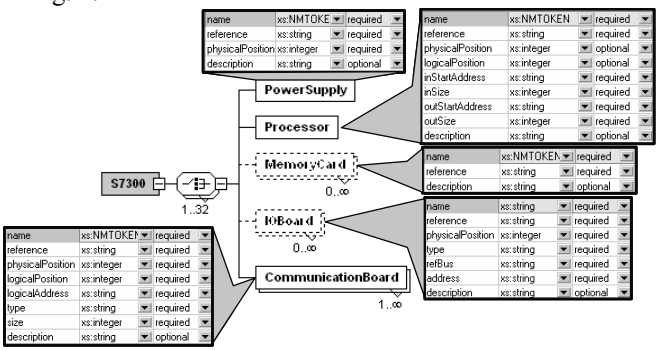

Fig. 4. Hardware templates for the Siemens S7300 family of PLCs

Finally, Fig. 5 illustrates an example of an IPMCS application that follows the proposed grammar (in order to

make it more readable, *Enhanced Grid view* offered by XMLSPY™ tool is used). This XML instance contains the three parts of an application model commented above.

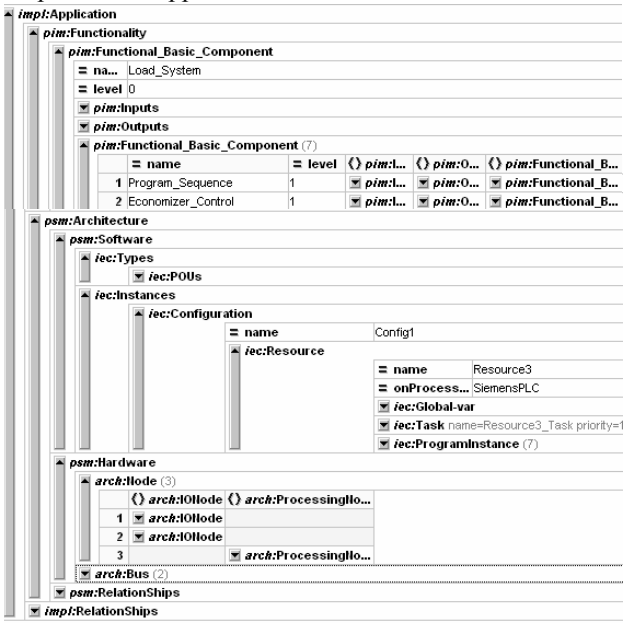

Fig. 5. IPMCS application example

B. Integration of a Tool within the Framework

This sub-section details the mechanisms that allow integrating a tool into the framework. Generally speaking, the tools generate / consume part of the model information and probably, expressed in different form. In this sense, the proposed framework must provide mechanisms that allow filtering, processing and adapting information from the tool to the model and vice versa.

First of all, it is necessary to do an exhaustive study of the tool *Application Program Interface* (API) as well as the type and format of the information the tool generates / consumes.

Depending on the level of processing and transformation needed, there are different XML technologies that can be used for implementing the tool integration. The XML *stylesheet* technology can be used to filter and transform the information between two different grammars, i.e. they allow transforming a XML document following a formal grammar into a XML document following other. This can be used when the tool API is XML-based. On the other hand, if the tool API is not based on XML, the *Simple API for XML* (SAX) [18] or *Document Object Model* (DOM) [19] should be used. They are very useful for getting information from a XML file and setting it to the tool by means of its API or for getting information from the tool and setting it into a XML file. In this form, although the tool API is not based on XML technologies, this tool can be integrated in the proposed framework. Note that it is also possible to use the combination of both technologies (*stylesheets* and SAX/DOM) in order to make easy the maintenance of the COTS tool version.

In order to illustrate this filtering and adaptation process, let us detail how to integrate the Siemens *"Administrator Simatic"* PLC programming tool, that does not follow the software model of the IEC 61131-3 standard. Thus, it is necessary to transform the standard software model to the proprietary software model. This can be achieved in two steps:

- Defining a XML schema for the proprietary software model (see Fig. 6).

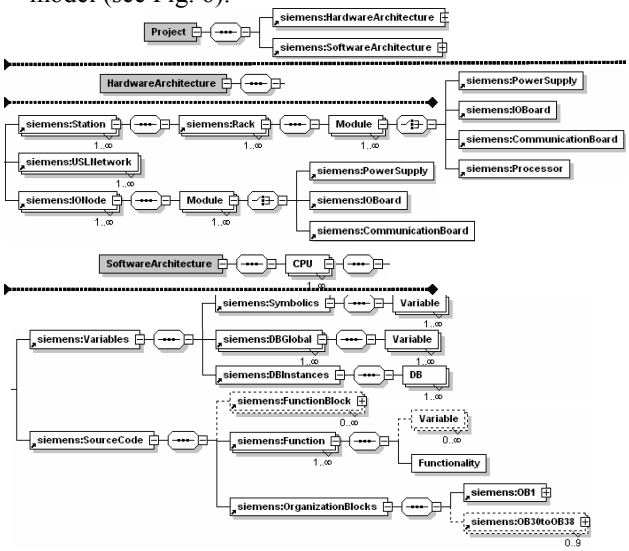

Fig. 6. Simatic formal grammar

- Developing a XML *stylesheet* that extracts the software architecture of the model XML file and transforms it to a XML file following the proprietary grammar.

On the other hand, as this tool does not offer and XML-based API, it is also necessary to develop an application that:

- Takes the proprietary architecture from the XML file, making use of DOM or SAX technology.
- Generates the automation project making use of the tool API.

Fig. 7 illustrates the necessary steps to integrate *Administrator Simatic*.

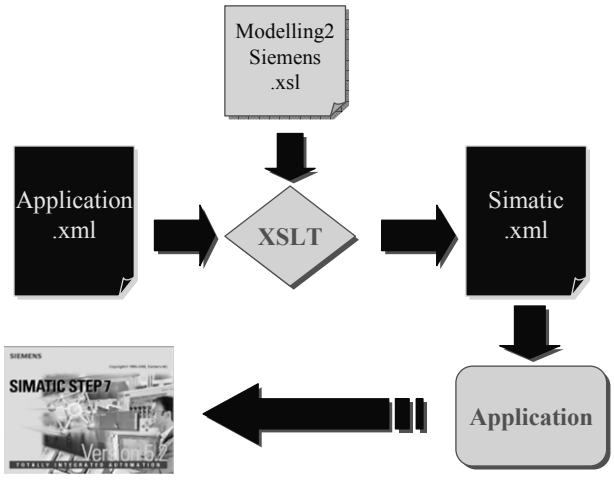

Fig. 7. Simatic automation project generation

The steps commented above should be followed to integrate any other tool in a framework.

IV. A FRAMEWORK FOR IPMCS

In this section the framework for IPMCs developed within the European project FLEXICON IST-2001-37269 is described. The general goal of the project is to develop methodologies that enable Commercial Off-The-Shelf (COTS) tools integration for the design and deployment of Distributed Control Systems (DCS) with high degree of flexibility, dependability and re-usability. This project specifically addresses the development of the capability to produce open, high performance, dependable, distributed fault tolerant systems in reduced timescales and at lower cost.

Within the toolset, the different phases of the development cycle are addressed using the appropriate tools. Fig. 8 illustrates the different tools involved during the application development cycle. In particular, the toolset integrates a set of COTS tools: ARTiSAN RtS UML tool as the modeling tool, and ISaGRAF Enhanced / SIMATIC as the PLC programming tool. On the other hand, tools specifically developed for the final user company are also integrated, such as a Through-Life Cost Model (TLCM) tool and a Reliability Model (RM) that are used to select the HW components. The toolset also integrates a co-simulation framework developed within the project [20] that closes the loop between the process model in Simulink and the control system model in ISaGRAF/Simatic Administrator.

As commented above, during the first phase, the functional and non-functional requirements are identified.

During the Analysis phase different type of analysis and simulations are performed, using the appropriate engineering tools. In this toolset, three main tools have been used: the Matlab-Simulink™ environment to model the process dynamic and perform simulations, the PLC programming tools (ISaGRAF Enhanced™ and Simatic Administrator™) for programming basic control functions and the Through-Life-Cost Modeling (TLCM) and Reliability Modeling (RM) Tools to select the most adequate HW architecture.

In summary, the significant process variables are identified (controlled, manipulated, disturbances), process models are obtained and control strategies are decided in order to meet the functional requirements. The selection of the needed instrumentation is also decided as well as the need of networks depending on the number of I/O signals and the physical distribution of the process. Thus, from this phase, a high level design of the control system emerges, as well as, the needs of distribution.

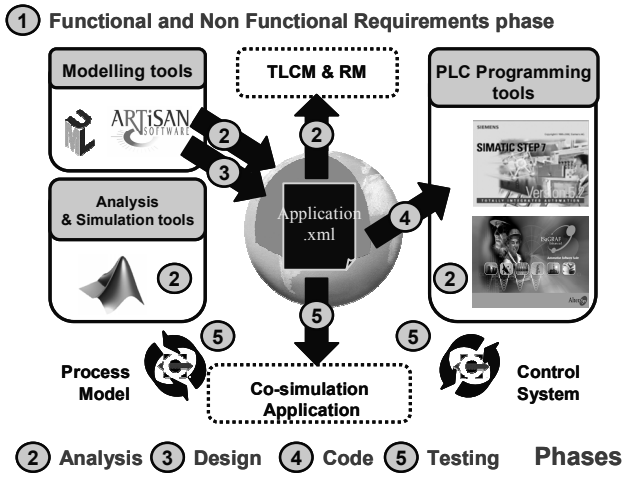

Fig. 8. The FLEXICON toolset for IPMCS

During the system design phase, ARTiSAN RtS™ UML tool is used. In order to guide non-expert UML users in the definition of the system views (functionality, hardware and software architectures and mappings between them), three UML profiles have been defined that group a set of stereotypes and tagged values that allow defining the components and connections of each part [21]. This phase provides a coherent and consistent model, developed using XML technologies [12].

During the Coding Phase, the source code of the programs within the resources of each configuration is automatically generated. This source code is generated in Structured Text (ST) language, as defined by the IEC 61131-3 standard. The last part of this phase is to automatically generate the automation project for the PLC programming tool, following the software architecture modeled (section 2B describes the necessary steps).

Usually, the detailed design of the previous phase is developed incrementally by coding parts of the application, performing unitary integration and validation tests. Thus, coding and testing phases are used iteratively for achieving control system validation. To do this, the toolset offers a co-simulation framework [20] that synchronizes the co-simulation between the model of the process and the control system, closing the loop between them. This is illustrated in Fig. 9.

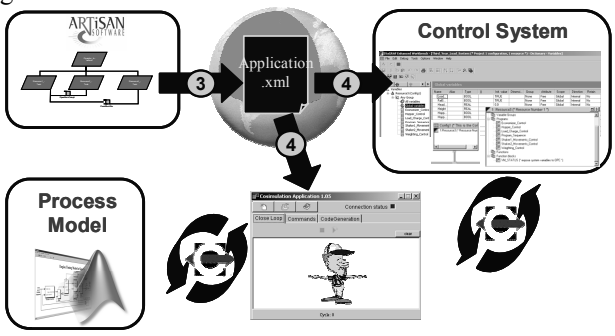

Fig. 9. Coding and testing phases

The co-simulation application also allows injecting process faults for validating the control system under different process operations.

The co-simulation framework gets the information from the *Application.xml* file related to the cyclic and timed resources as well as the information exchanged between the process model and the control system. This information is needed in order to compute the time instants in which the data exchange should be performed.

During the testing phase, it is also possible to test the involved hardware (inputs and outputs nodes, bus segments, PLCs and/or other controllers, etc) using co-simulation with Hardware In the Loop (HIL), as illustrated in Fig. 10.

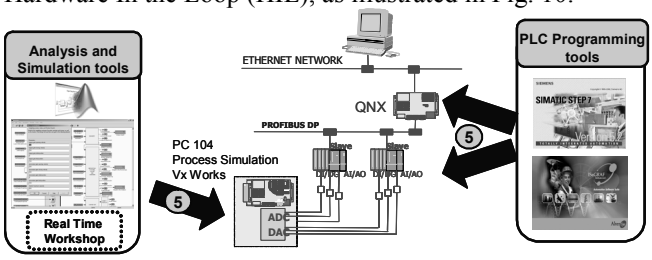

Fig. 10. Coding and Testing: co-simulation with HIL

The co-simulation with HIL allows not only to check the hardware selected for implementing the control system, but also to inject process faults. In this framework, the toolbox Real Time Workshop™ [22] has been used for generating the real-time code of the process model under VxWorks operating system.

V. CONCLUSION

Integration and collaboration of the software tools involved during the development cycle of distributed industrial control systems is a key issue in the design of industrial applications. Usually the tools are heterogeneous not only with respect to the domain of expertise they belong to but with respect to the hardware and software platform in which they execute. Thus, the major problem to make them collaborate is to adapt the information about the system from the semantic and format of one tool to the other. This paper proposes a tool integration and collaboration framework based on a formal model that contains separate views of the application: functionality, hardware architecture and software architecture. The framework provides the necessary mechanisms for both adapting the information generated by a tool the application model and extracting information from the model and adapting it to the tool semantics. XML technologies have been proved to be very powerful to achieve these goals. XML schemas and Schematron rules are used to formally describe the model of the distributed system and to perform model consistence and data coherence analysis. XML stylesheets are used for filtering, processing and adapting information between the application model and tools. Thus, making use of these technologies, the framework can be customized to integrate a particular set of tools. A toolset for IPMCS has been developed and it has been validated in industrial application.

ACKNOWLEDGMENT

This work has been supported in part by EU-IST programmed under project IST-2001-37269 and by MCYT&FEDER under project DPI 2003-2399.

REFERENCES

[1] Lewis, R.W, *Programming Industrial Control Systems using IEC 1131-3*,.IEE Control Engineering, 1998.
[2] Extessy AG. (2003) Available www.extessy.com
[3] The MathWorks (2002), *Using Matlab version 6.5*
[4] Artisan (2002). Manual of ARTiSAN Real-Time Studio, Version 4.2, ARTiSAN Software tools.
[5] BarDyne, Inc: Hidraulic and Fluid Power Experts (2003) Available www.bardyne.com
[6] TNI Software: CosiMate (2004) Available www.tni-world.com
[7] George N. "Cosimulation of an Automotive Control System using ADAMS and Xmath". *International ADAMS User Conference. Utrecht, Netherlands*. 1998.
[8] ISaGRAF Enhanced- Workbench 2.4 Altersys. 2002.
[9] SIMATIC. 1996.
[10] PLCopen, 2003. Available: http://www.plcopen.org/
[11] PLCopen, "TC6 for XML", 2004. Available http://www.plcopen.org/TC6/XML_Intro.htm
[12] Marcos M, Estévez E. "Formal modelling of Industrial Distributed Control Systems". *16th IFAC World Congress in Prague*.2005. Submitted for publication.
[13] W3C, *XML (2003),* Available http://www.w3.org/XML/
[14] van der Vlist E., *XML Schema,*. O'REILLY, 2002.
[15] Doug Tidwell, *XSLT,* .O'REILLY, 2001.
[16] Topologi, *schematron* (2001), Available http://www.ascc.net/xml/schematron/
[17] Karl-Heinz J., Tiegelkamp M. *IEC 61131-3: Programming Industrial Automation Systems.* Springer-Verlag Berling Heidelberg. 2001.
[18] SAX, Available http://www.saxproject.org
[19] DOM, Availabe http://www.w3.org/DOM
[20] Marcos M., Gangoiti U., Orive D., Estévez E., Calvo S. Barandiarán J. "Design and Validation of Industrial Distributed Control Systems". *43rd IEEE Conference on Decision an Control, Nassau, Bahamas*. 2004.
[21] Marcos M., Estévez E., Gangoiti U., Sarachaga I., Barandiarán J. "UML Modelling of Industrial Distributed Control Systems", Sixth Portuguese Conference on Automatic Control CONTROLO 2004, Faro, Portugal. 2004.
[22] Real Time Workshop (2005), Available http://www.mathworks.com/products/rtw/