CONTROL FUNCTIONS DEVELOPMENT FOR DISTRIBUTED AUTOMATION SYSTEMS USING THE TORERO APPROACH

Luca Ferrarini[1], Carlo Veber[1], Christian Schwab[2], Marcus Tangermann[2], Aggeliki Prayati[3]

[1] Politecnico di Milano, Dip.di Elettronica e Informazione, P.zza L. da Vinci 32, 20133 Milano, Italy, {ferrarin, veber}@elet.polimi.it
[2] University of Magdeburg, Center Distributed Systems CVS@IAF, Universitaetsplatz 2, 39106 Magdeburg, Germany, {christian.schwab, marcus.tangermann}@mb.uni-magdeburg.de
[3] University of Patras, Dept.of Electrical & Computer Engineering, Campus of Rio, 26500 Patras, Greece, prayati@ee.upatras.gr

Abstract: The paper presents the design and implementation of an innovative Integrated Development Environment, developed within the on-going European Research Project TORERO, to model and design reusable distributed control systems (DCS). This project proposes an architecture which relies on a development environment based on the Eclipse tool and the emerging standard IEC 61499 and on a suitable control device where a portion of the distributed control application can be executed on a specially developed runtime environment.

Keywords: Distributed Computer Control Systems, Standards, Functional Blocks, Control System Design Methodology, Industry Automation.

1. INTRODUCTION

The control and automation of modern manufacturing systems is characterized by a strong push towards modularization and autonomy, which implies the distribution of "intelligence" (that is control laws, hardware and software control components) into the controlled plant. In addition, to enhance the overall system performance and to integrate automation level with management level, there is a natural trend to include methods and tools of the Information and Communication Technologies down into the automation systems (Heck B.S., 2003). On the other hand, control components of manufacturing systems have not experienced the same trend to modularization and standardization as the other plant components, most notably the mechanical and the electrical ones. Such a delay has the consequence of increasing the various costs involved in the design, implementation, testing, installation and maintenance of the overall control system, and reducing the possibilities to gain real flexibility, real reconfiguration and reuse of control solutions (Ferrarini L., 2003a), (Ferrarini L. 2003b). Clearly, this requires new modeling paradigms able to capture the overall system description and help the translation of the more and more strict control specification into control design and implementation. Traditional design models and approaches used by control engineers are based on low-level modeling paradigms (often programming languages), centralized approaches, proprietary tools and solutions (Prayati A., 2003). Finally, the possibility of formal analysis of design consistency and fulfilment of the specifications is reduced to sporadic testing or debugging.

In the recent technical literature, some proposals can be found on possible solutions to the problem of designing distributed control systems (DCSs) and the current lack of interoperability. A first attempt has been performed in the OSACA project (Open System Architecture for Controls within Automation Systems) in '90s, whose main goal was to specify a system software architecture for open control systems, which is manufacturer independent (OSACA, 2004). Unfortunately, the basic ideas there contained did not further developed into suitable standards or tools. In the robotic field, similar efforts

have been carried out. Although robotic applications have their own specific features, the concepts of hierarchy, modularisation, abstraction from the low level systems, separation of functions from code, and finally, the adoption of "open" schemes are quite similar to more general automation systems.

The paper addresses the problem of modeling and design of (DCSs) and in particular it focuses on the design and the implementation of an Integrated Development Environment. The work is drawn from the on-going European Project TORERO (TORERO, 2004). As opposed to the above mentioned approaches, such a project does not proposes software infrastructures, like middleware services (see e.g. CORBA ones) or predefined modules, but on the contrary proposes a generic standardized automation device, an abstraction of communication services and a high level formalism based on IEC 61499 international standard (Holobloc.com, 2004). Its key points include the automatization of all the design steps, from the integration of plug-and-participate mechanisms, automatic distribution of function blocks to the devices, automatic generation of Java code, automatic weaving on the most common field busses and automation protocols.

The paper is organized as follows. In Sect. 2 the approach proposed by the TORERO consortium for an integrated environment that allows the design and implementation of distributed control systems is described. In Sect. 3, the software tools for the implementation of an integrated development environment following the chosen approach are presented. The two subsequent sections describe the design of a portion of this development environment that allows the design and the implementation of distributed control applications: Sect. 4 presents the overall design workflow while, in Sect. 5, the main elements of such an environment are described.

2. TOTAL LIFE-CYCLE APPROACH: TORERO

The European research project TORERO (TORERO, 2004) aims at defining an integrated distributed non-hierarchical environment that will allow the distribution of both control applications and management functionalities, and will allow the seamless functioning of intelligent mechatronic devices. This happens in order to allow for the maximum possible re-use and self-configuration throughout the life time of the DCS.

The control system in TORERO is based on the definition of a new device type (TORERO Device, TD). The TORERO project develops a new methodology, which takes these actors in account against the background of ordinary design and development, software updates and changes, and bug fixes.

Deriving from these considerations, three different processes have been identified: The engineering, the re-engineering and the maintenance process. Both engineering and re-engineering processes are understood as major changes to the system, whereas maintenance causes minor changes and can be performed with less effort. Both engineering and re-engineering process can be mapped to an unified approach, which takes the necessities of a fully DCS into account (see Fig.1).

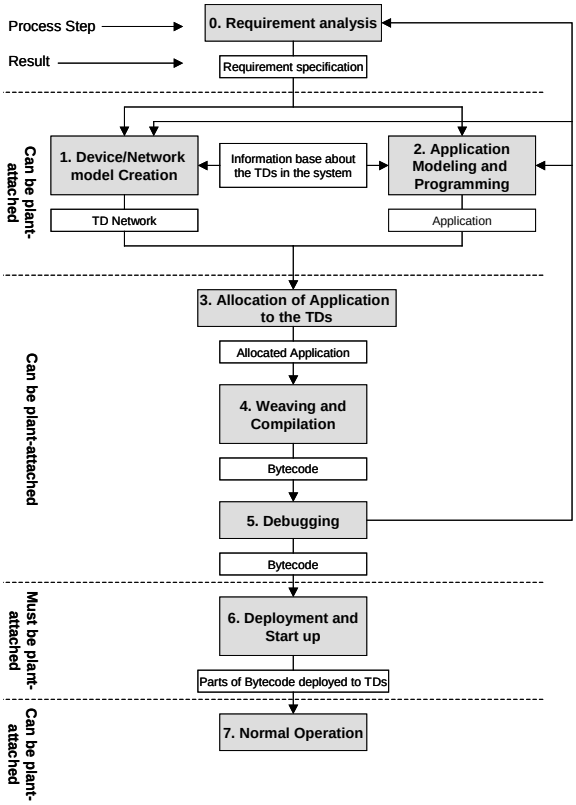

Fig.1. (Re-)Engineering process defined in TORERO

This (re-) engineering process allows for the re-use of software regarding both application set and specific application by making the application design independent from the proposed physical architecture, as allocation and weaving of communication code are performed later in the process (steps 3 and 4). During allocation, the allocation of the different function blocks of the application to the control devices is performed. In this process, communication needs and device resources together with real-time aspects are taken into account. Following the allocation, the appropriate communication code is woven into the application (Schwab C., 2004). Also, the re-use of software at device level is allowed by the possibility to incorporate device specific code during step 2.

The TORERO approach allows a clear separation between functional design (the user is set free to choose the best one for his application) and implementation which is distributed, compliant with IEC 61499 and assisted through an automatic allocation of FB to devices, an automatic insertion of com-munication FB and an automatic translation of FB application into Java language. As for the last point, specific novel methods of translation have been conceived and tested in order to obtain a Java control code more suitable and apt for industrial automation applications, as shown in (Ferrarini L., 2004). This approach in its entirety means a novel approach in opposition to current state of the art (Holobloc.com, 2004), which always only considers single aspects of the entire (re-) engineering process.

As consequence to the above stated, TORERO defines the smallest part of a control system that can interact within such an environment - the "TORERO device" – which will comprise an architecture governing the control application deployment in the device, as well as a management part determining the device behaviour on the basis of information communicated to the device and its state (see Fig. 2). Consequently, a TORERO device comprises Control Application Objects relevant to the part of the overall control application that is deployed on it, I/O connections related to the actual physical connections of e.g. sensing and actuating elements that the device comprises, and Device functionalities such as Forced Code Deployment, Code Download both in a local scale or from the net, Diagnostic Information, Version Control, etc.

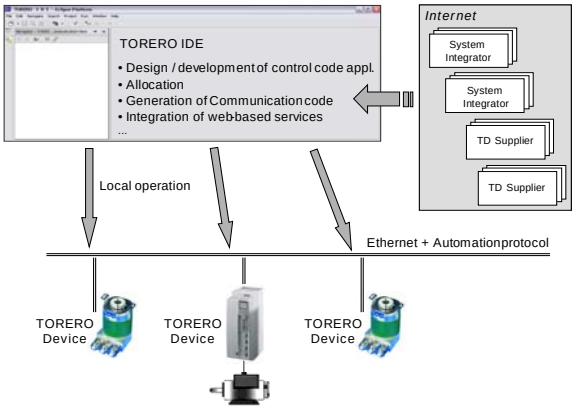

Fig. 2. The TORERO overall system architecture.

The design and development of control applications and the integration of such TORERO devices requires the use of a TIDE (TORERO Integrated Development Environment). The TIDE is an off-line tool that includes several functionalities such as allocation of control applications to TORERO devices, support of their configuration, deployment of the code and configuration to the devices, validation/verification support, version control support, documentation support, support of the reuse/import of previously developed control functions, specification of the communication between the TORERO devices, translation of the control application to device specific languages and support of the design of graphical user interfaces (GUI) of the application.
Fig. 3 shows the architectural design of the TORERO Device and how the different elements within the device are mapped within the TIDE. In this picture it can be seen amongst others that TORERO differentiates between device functions (which are typical to the device) and application functions and that TORERO proposes Java as control function execution language. Regarding the device functions it can also be stated that they can be delivered as OS based (meaning there is no possibility to change these functions) and Java-based which means they can be changed by the application builder.

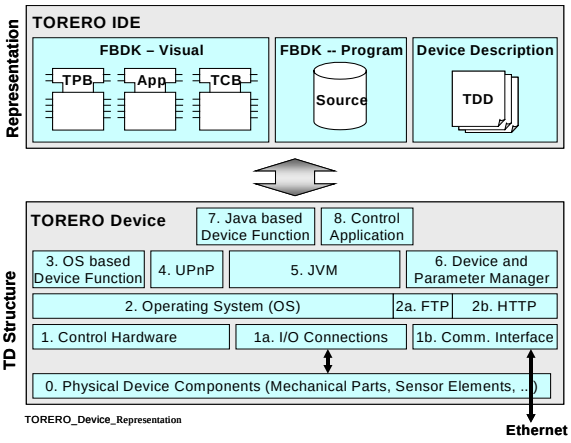

Fig. 3. TORERO IDE and TORERO Device model.

3. SOFTWARE TOOLS FOR TIDE

As mentioned above, the TORERO Integrated Development Environment (TIDE) will be developed using Eclipse as programming tool and as runtime environment (www.eclipse.org). Eclipse is an open source platform for tools, which can be used in a variety of applications. Based on standard generic components such as a navigator view, a text editor, a task view, Eclipse provides a powerful plug-in concept to adapt it to the specific needs of the user. Main plug-ins of the developed TORERO IDE are:

- TIDE Device Model (see section 5.1),
- PnP Plug-in using the Universal Plug-and-Play stack (Cohen, 2000),
- Allocation plug-in (see section 5.3),
- Weaving plug-in (see section 5.4),
- Control Application Editor as an external tool (see section 4).

The Eclipse tool itself and the plug-ins are programmed in Java and a so-called manifest file declares the interconnections between the plug-ins.

The actual control application will also be programmed in Java. As the TORERO architecture favours the concepts of distributed automation, it requires the download of distributed control code to the TORERO devices. It is also required that the TORERO architecture works with devices of different manufacturers that may contain different processing platforms. This leads to the necessity of having a hardware platform independent, standardised software platform inside the TD. Such a platform is provided by Java, which runs as byte code on a Java Virtual Machine hosted by the TD.
As strong restrictions of embedded devices to the processing hardware apply, a simplified JVM may be used like it is provided by the KVM (Kilo Virtual Machine) (KVM, 2004), which was developed by Sun to be used in handheld devices and is best suited for the needs of embedded devices. It does not support all commands of a full JVM though, but it needs less computing power and memory resources. The KVM is supported by the CLDC (Connected Limited Device Configuration) configuration, especially designed for resource-constrained devices.

As for the Control Application Editor instead of developing an entire editor inside Eclipse, the preference has been given to an existing free tool named FBDK (Holobloc.com), compliant with the standard IEC 61499. Such a tool has been developed by Rockwell Automation for demonstration purposes, with the following features: editing of basic, composite, communication function blocks and of resource, device and the overall system configuration; exchanging data in XML format; generation of java code; simple graphic interface library; simulation.
As for the generation of executable code the IEC 61499 does not provide any formal specification. In (Ferrarini L., 2004) several approaches for code generation are shown and analysed and in particular it is shown that the FBDK (version of Sept. 2003) suffers from some drawbacks related to event propagation that introduces unnecessary delays in the execution of function blocks activated by a single event. To overcome such drawbacks and improve the overall performances of the control application code, TORERO proposes an innovative thread-based approach as described in the above paper.

4. TIDE DESIGN FOR CONTROL APPLICATIONS

In this section, the focus is on the design of the TIDE functionalities specifically devoted to the design and implementation of a distributed control application. The final aim is to automate as much as possible all the design and implementation steps in the engineering process. To do so and exploit the functionalities of Eclipse, FBDK and KVM, the Data Flow Diagram of Fig. 4 has been conceived.
Starting from the descriptors of the TDs (TDDs, TD Descriptions) contained in a suitable database (see the datum D1 of Fig. 1), an implementation model of the TDs of the system is instantiated in the TIDE by the TIDE Device Model plug-in. This model also contains the java classes representing the proxy functionalities of the TDs connected. The corresponding XML format (IEC 61499 compliant) of these functionalities are generated by the XML generator (provided by the XML Parser plug-in). The XML files can be used in the external application editor FBDK, where the control application is designed. The output of such tool is constituted by a IEC 61499-compliant XML code used by the Application Importer (provided by the XML Parser plug-in) to generate the java (TORERO compliant) classes related to the FBs of the given application. Such an application also extracts the control application (FB instances and their connections) and the FBs parameters relevant to the Allocation Algorithm, whose plug-in maps the application FBs to the TDs connected. Starting from the topology of the control application given by the XML Parser plug-in and the given mapping, the last plug-in, Weaving and Compilation, automatically modifies this topology by inserting suitable Communication FBs and prepares the code executable by each TD. Finally, installed on the TD there is a JVM and the Run-Time Environment (RTE) which is able to execute the downloaded code for the device.

5. TIDE PLUG-INS AND RTE

The present section describes the main plug-ins and the RTE that allow the design and implementation of the distributed control application according to the TORERO approach.

5.1 TIDE Device Model

The TORERO Device Model is a representation for a TD within the TIDE and acts as a database for all actions performed by the TIDE during the (re-) engineering and maintenance process.

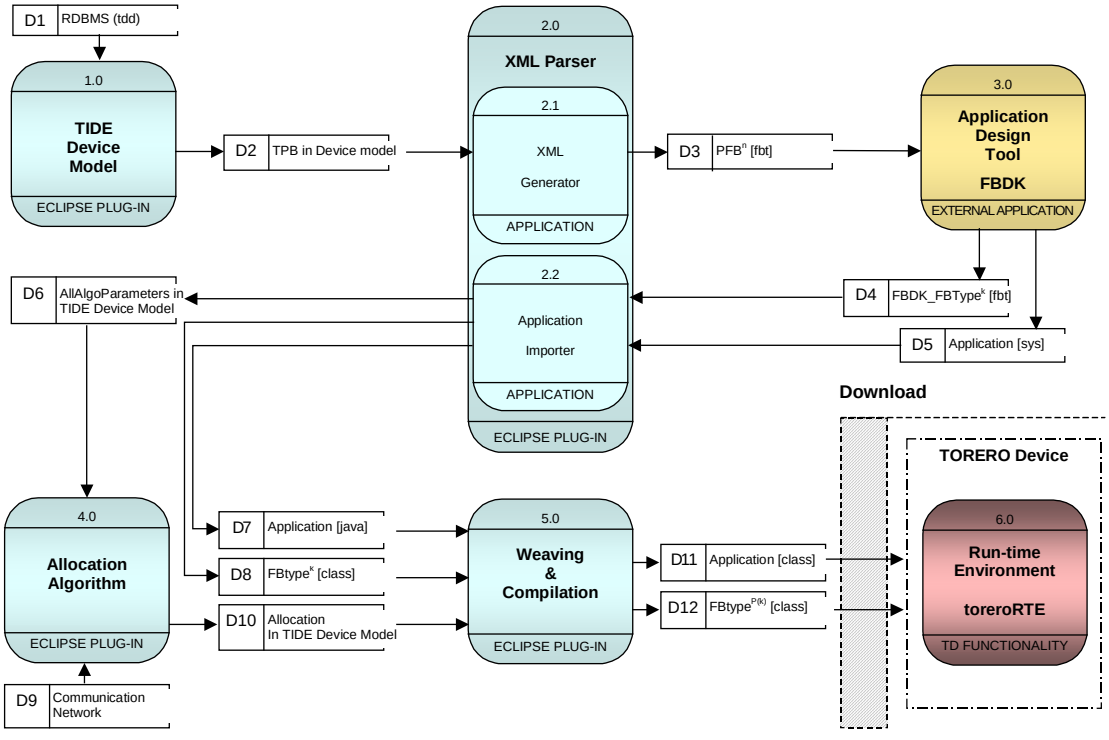

Fig. 4. TIDE Data Flow Diagram for the engineering process.

It can be divided into two main parts:

- Device Model specific classes, and
- Function Block specific classes.

The first group of classes contains device specific information such as:

- basic information of the TD (vendor, serial number)
- control hardware (processor type and vendor, storage size),
- operating system (vendor version),
- device functions (OS based and Java based),
- Java Virtual Machine,
- Communication Interface of the TD (Ethernet interface),
- physical device components (e.g. sensor elements),
- and the configuration (e.g. IP address).

The Function Block specific classes describe all Function Blocks installed on the TD, whereas the Function Blocks for the control application, the TORERO Communication Blocks, and the TORERO Proxy Blocks are considered. In particular, these classes encapsulate the elements of IEC 61499 Function Blocks such as the Algorithm, Execution Control Chart, event and data connections.

The TIDE Device Model will be generated automatically within the TIDE by extracting all relevant data from the TDD transferred from the TD to the TIDE. The TDD is an XML file in which the TORERO specific description extends the XDDML (FDCML, 2004) which itself extends the XML file used by the Universal Plug-and-Play stack. The following extensions to the mentioned description languages are necessary to cover all data describing the TORERO device and the installed software: communication aspects of the TD, OS, JVM, and application process.

5.2 XML Parser

This plug-in is composed of two applications.

XML Generator. It creates the XML IEC 61499-compliant code of the Proxy Blocks stored in the TIDE Device Model.

Application Importer. The chosen execution model is the thread-based one described in (Ferrarini L., 2004). To implement it, a few classes related to the event propagation have been designed and this application generates and compiles, for each FB type generated by FBDK (in XML format), the corresponding java class (TORERO compliant) and generates a java file for the whole control application. This application extracts also information about the complexity of a FB execution and the time constraints associated to event connection among FBs.

5.3 Allocation Algorithm

For the given heterogeneous system characteristics and distributed application requirements, an allocation and scheduling algorithm has been proposed (Prayati A., 2004). The allocation algorithm is applied for the FBs allocation to the system devices. FBs are considered as tasks that need to be assigned to the system resources. The proposed approach follows the basic guidelines of existing allocation algorithms (Hou C., 1997)(Jonsson J., 1999) and proposes a hybrid technique where first clusters of tasks are formed and then the Branch & Bound (B&B) technique is applied on clusters of tasks instead of one single task. In that way, the vertexes of the B&B search tree represent partial assignments of clusters of FBs to the devices resources. By forming clusters of tasks before applying the B&B algorithm, the search space is considerably reduced, while at the same time the drawback of the clustering technique is eliminated, since the final allocation scheme is based on mathematical calculations after scheduling, when the allocation is already known and then the real execution time values are used. The allocation algorithm composes of three basic steps:

- priority assignment, where the tasks are assigned a priority with respect to the order they will be considered during the assignment procedure in the next allocation step. Task priority expresses the time criticality of the task's algorithm execution.
- clustering of tasks, where tasks are grouped and considered as super-tasks (clusters), under the objective that communication time among tasks is the minimum possible one.
- B&B is performed on the clusters of tasks in order to find the optimal task assignment to the system TDs, with respect to the imposed timing constraints. This latter is handled by scheduling all the alternative cluster assignments to the given TDs.

Time constraints are checked for every inter-task communication edge and execution time, with respect to latency and bandwidth of the HRC IAONA classification (IAONA, 2003). If the constraints cannot be met, the virtual values for the priority assignment are adjusted and the algorithm is repeated from the beginning, until a feasible allocation scheme is generated.

5.4 Weaving and Compilation

After the allocation of the control application to the TDs, Step 4 of the engineering process deals with the automatic generation and adding of communication related code (aspect code) into the control application code (component code) using the aspect oriented programming (AOP) language AspectJ (Tangermann M., 2003) and with respect to the underlying automation protocol. The aspect code can consist of several aspect programs, each of which implements a specific aspect (e.g. different automation protocols used in the system). Crosscutting lies at the heart of aspects. Modular units of decomposition are organised into clear hierarchies, whereas aspects crosscut such hierarchies. Join points act as the location where the aspect code is inserted into the component code, where the crosscut is. The Aspect Weaver is the core component of AOP, which takes the aspect code and the component code, finds join points and weaves them all together to form a single entity.

This approach for the generation of communication related code gives the following benefits:

- generic approach for the communication interface,

- possibility to use different communication protocols with no change in the control application code, e.g. time-triggered (Ethernet Powerlink) or event-triggered (Modbus/TCP) protocols (Schwab, 2004),
- integration of local (TD intern) and remote (via a communication infrastructure) access without adaptation of the control application code.

5.5 Run-Time Environment (RTE).

The designed application is downloaded into TDs in the form of java byte code. The Run-time Environment is a portion of the native software of a TD, which is able to execute the native device functions and the control application. In particular, to start a control application, it simply initialises the system instantiating one thread for each FB-class and successively "drawing" the event connections using the methods of these classes. After that an application can be started, stopped or restarted using the corresponding methods of each FB-class.

6. CONCLUSIONS

In the paper the design and the implementation of a development environment for the design and the implementation of control applications for distributed control systems is presented. Such an environment, called TIDE, has been developed within the TORERO European project, integrating the Eclipse platform with FBDK, a prototype tool to edit and simulate a control function application compliant with IEC 61499. The TIDE is able to automatically allocate control function blocks to the control devices, according to user-defined real-time constraints, and generate java code for distributed control applications running on micro-VM. Future work includes the addition of functionalities like closed-loop distributed simulation with the semi-automatic generation of the simulation model of the process under control.

REFERENCES

Eclipse.org (2004).Available:http://www.eclipse.org.

FDCML (2004). Available: www.fdcml.org.

Ferrarini L. and Veber C. (2004). Implementation approaches for the execution model of IEC 61499 applications, *INDIN'04 – 2nd IEEE International Conference on Industrial Informatics*, June 24-26, Berlin, Germany.

Ferrarini L., Veber C. and Fogliazza G. (2003a). Modelling, Design and Implementation of Machining Centers Control Functions with Object-Oriented Techniques, *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, Vol. II, pp. 1037-1042, July 20-24, Kobe, Japan.

Ferrarini L., Veber C. and Lorentz K. (2003b). A case study for modelling and design of distributed automation systems, *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, Vol. II, pp. 1043-1048, July 20-24, 2003, Kobe, Japan.

Heck B. S., Wills L. M. and Vachtsevanos G. J. (2003). Software Technology for Implementing Reusable, Distributed Control Systems, *IEEE Control Systems Magazine*, Feb., pp. 21-35.

Holobloc.com (2004). Function Block-Based, Holonic Systems Technology. Available: www.holobloc.com.

Hou C., Shin K. (1997). Allocation of Periodic Task Modules with Precedence and Deadline Constraints in Distributed Real-Time Systems, *IEEE Transactions on computer*, vol. 46, n.12.

IAONA N.N. (2003). Joint technical working group Hard Real-time, first draft for real-time classification, unpublished working group paper. Available: www.iaona-eu.org.

International Electrotechnical Commission, Techincal Committee 65 Working Group 6, IEC-TC65 (2000). Function Blocks for Industrial-Process Measurements and Control System, Parte I, Architecture, 2000, PAS.

Jobling C. P., Grant P. W., Barker H. A. and Townsend P. (1994). Object-oriented programming in control system design: A survey, *Automatica*, 30, n° 8, pp. 1221-1261.

Jonsson J. (1999). A Robust Adaptive Metric for Deadline Assignment in Heterogeneous Distributed Real-Time Systems, *Symposium on Parallel and Distributed Processing*.

KVM (2004). Kilo Virtual Machine, http://java.sun.com/ products/cldc/.

OSACA (2004). Open System Architecture for Controls within Automation Systems. Available: http://www.osaca.org.

Prayati A., Kalogeras A., Lorentz K. and Koubias S. (2003). Engineering Tools to Support Interoperability in the Development and Maintenance of Heterogeneous Distributed Real-Time Control Systems. *Proceedings ADSN – International Workshop on Assurance in Distributed Systems and Networks*. Providence (RI), USA.

Prayati A., Koulamas C., Koubias S., Papadopoulos G. (2004). A methodology for the development of distributed real-time control applications with focus on task allocation in heterogeneous systems, *IEEE trans. on industrial electronics*.

Schwab C., Tangermann M., Lüder A., Ferrarini L. and Fogliazza G. (2004). Mapping of IEC 61499 Function Blocks to automation protocols in the TORERO project, INDIN'04 – 2nd IEEE International Conference on Industrial Informatics, June 24-26, Berlin, Germany.

Tangermann M., C. Schwab, A. Kalogeras, K, Lorentz, A. Prayati (2003). Aspect-Orientation of Control Application Code for Distributed Automation Systems: The TORERO Approach, JTRES 2003 – OTM Confederated International Workshops, Java for Real-Time and Embedded Systems, November, Catania, Italy. LNCS 2889, Springer-Verlag, Berlin, Heidelberg.

TORERO Project (2004), IST-2001-37573,Total life cycle web-integrated control, funded by the European Community under the IST Programme (1998-2002). Available: www.uni-magdeburg.de/iaf/cvs/torero.