Goran Ferenc
Zoran Dimić
Maja Lutovac
Jelena Vidaković
Vladimir Kvrgić

ISSN 1333-1124

OPEN ARCHITECTURE PLATFORMS FOR THE CONTROL OF ROBOTIC SYSTEMS AND A PROPOSED REFERENCE ARCHITECTURE MODEL

UDC 004.45:004.896

Summary

This paper presents advantages of using open architecture for the real-time control of robot manipulators, parallel kinematics machine tools and other multi-axis machining systems. In order to increase their competitiveness, companies need to follow the global economy requirements. The constant incorporation of new technologies into existing controllers and reduction in the development time and costs are the main objectives. An open architecture control (OAC) concept appears as a solution to deal with these requirements. This article explains the rationale for the development of OAC systems, presents the major international activities which propose various approaches to OACs and a series of controllers that have been developed using this design philosophy at the Lola Institute.

Key words: *open architecture control, robot manipulators, machine tools*

1. Introduction

Device-oriented, dedicated and heterogeneous systems where the application software, the system software, and the hardware are tightly coupled dominated in the past. This approach led to complexity and inflexibility, long development time, high design costs and very difficult way to introduce new functionalities into such systems.

Due to the high innovation speed in the hardware and software technologies, it becomes necessary for companies to develop hardware-independent software as far as possible in order to stay competitive [1]. Companies have to take into account easy future system upgrades, modifications and extensions. These requirements are met using an open architecture-oriented structure.

Significant efforts have been made to maintain and further develop the products according to these requirements. The early nineties are marked as the beginning years of initiatives for enabling control vendors, machine tool builders, and end users to benefit more from modular and flexible production facilities. At the start of using the open architecture approach, the goal was oriented to customer requirements and an as-easy-as possible implementation of customer-specified controls. This related to open interfaces and configuration methods in a standardized, vendor neutral environment. Such systems were broadly accepted, which resulted in reducing costs and increasing flexibility. System integrators reused the software and implemented algorithms specified by the user, while users were able to design their own controls based on the given configurations.

Modern approaches favor PC-based solutions with a homogenous, standardized environment, where the structure is software-oriented and configurable due to defined interfaces and software platforms. New advanced functionality has to be continuously integrated into control systems and reconfigurable manufacturing units have to be created. Open control interfaces enable these features. Hardware and software separation allows to profit from the short innovation cycles of the semiconductor industry and information technology. With the possibility for reusing software components, the performance of the overall system enhances simply by upgrading the hardware platform [2].

According to the IEEE 1003.0 model, an open system is defined as a system which provides capabilities that enable properly implemented applications to run on a variety of platforms from multiple vendors, to interoperate with other system applications and to represent a consistent style of interaction with the user.

The openness of a controller is defined through:

- **Portability**, which indicates that application modules can be used on different platforms without any changes, while maintaining their capabilities.
- **Extendibility**, which implies that a varying number of application modules can be run on a platform without any conflicts.
- **Interoperability,** which indicates that application modules can work together in a consistent manner and can interchange data in a defined way.
- **Scalability**, which implies that the functionality of the application modules, performance, and the size of the hardware can be adapted depending on the user requirements.

According to the IEEE definition, an open control system must be:

- **Vendor-neutral**, which guarantees the independence of single proprietary interests.
- **Consensus-driven**, which means that it is controlled by a group of vendors and users (usually in the form of a user group or an interest group).
- **Standards-based,** which ensures a wide distribution in the form of standards.
- **Freely available**, which means it is free of charge to any interested party.

2. Major international activities

Many international organizations have done extensive research on the topic of open architectures around the world, but extra effort has been devoted to that topic in the USA, Germany and Japan. All these architectures have integrated equipment of several manufacturers and offer control solutions at a lower cost, while maintaining the same performance. The most significant activities and their main characteristics are presented in the following subsections.

2.1 OSEC

In December 1994, three machine tool builders, Toyoda Machine Works, Toshiba Machine, Yamazaki Mazak, two information system companies, IBM Japan, SML, and a controller builder, Mitsubishi Electric, established a project group, OSEC (Open system Environment for Controllers). The objective of the group was to develop an open architecture platform for numeric control equipment. Since then, several companies and institutions have joined OSEC. At one point, about twenty companies and organizations were promoting the project [3].

The idea for creating the OSEC architecture emerged from the need to provide end users, machine makers, control vendors, software vendors, system integrators and others with a standardized platform for industrial machine controllers which would enable them to add their own unique features to the industrial machines, and hence to promote the technical and commercial development of the industrial machines [2].

OSEC is focused only on the PC platform and Windows environment and it does not allow distributed control [3]. The purpose of creating a PC-based manufacturing equipment controllers of an open architecture is to develop a high-performance, low-cost and an easy-to-maintain control system. Emergence of PC-based open controllers has made the construction and operation of these machines more efficient since they will be integrated into a networking application and will exchange information with remote machines. When this architecture becomes open, manufacturers will be able to integrate their own control systems with various combinations of PCs and control subsystems based on it. Since the architecture is based on PCs, high quality software can be expected in the area of factory automation. A PC can act not only as a controller of equipment, but also as the basis of an information system for plant operation.

OSEC architecture is shown in Fig. 1. Reference model is structured in layers. The OSEC API (Application Programming Interface) is defined in the form of an interface protocol, which is used to exchange messages among controller software components representing the functionality and the real-time cycle. Programming interface between different layers is called Message Coordination Field. Each functional block can be encapsulated as an object, so it is not necessary to deal with how a functional block processes messages to it at the architecture level [3].

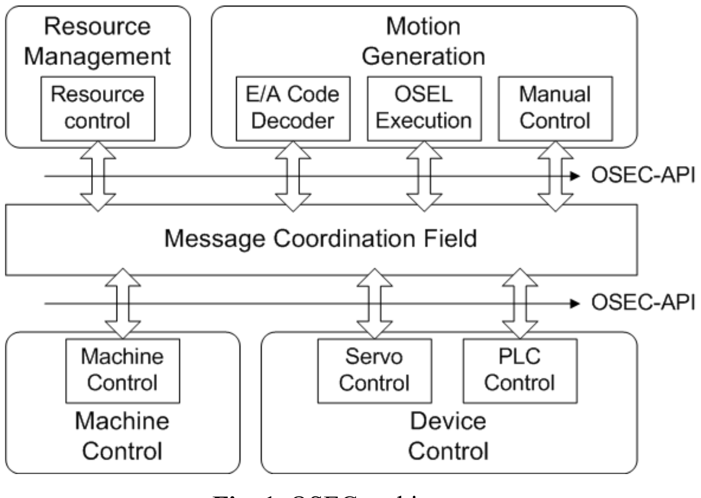

Fig. 1 OSEC architecture

OSEC architecture is abstract as a result of generalization to cover a wide range of control systems as much as possible. Although the structure of functional blocks can be defined uniquely by the OSEC architecture from a logical point of view, there are so many options for implementations. The following functions can be implemented: device driver, inter-process communication, installation mechanisms such as static library and DLL, hardware factors like selection of controller card and implementations of software modules added for execution control and/or monitoring of various software. In other words, the implementation model is not limited to a particular model and thus various ideas in the implementation model depending on the system size or its hardware implementation and/or utilization are made possible [3].

2.2 OMAC

OMAC (Organization for Machine Automation and Control) is formed as the "Open Modular Architecture Controls User's Group" to provide a global organization that supports machine automation and operational needs of manufacturing. It enables companies to work together and to [4]:

- Cooperate in finding common solutions for both technical and non-technical problems in the development, implementation and commercialization of the open, modular architecture control technology.
- Promote the open, modular architecture control development among control technology providers and its adoption among end users, OEMs (original equipment manufacturers) and system integrators.
- Provide responses to the users of an open, modular architecture, using the experience of software developers, hardware builders and OEMs in manufacturing applications.
- Enable collaboration with user groups worldwide to achieve common international technology guidelines.

The OMAC API adopted a component-based approach to achieve plug-and-play modularization, using interface classes to specify the API [2]. For distributed communication, the component-based technology uses proxy agents to handle method invocations that cross process boundaries. OMAC API contains different sizes and types of reusable plug-and-play components: component, module, and task - each with a unique Finite State Machine (FSM) model so that the component collaboration is performed in a familiar way. The component is a reusable piece of software that serves as a building block within an application, while the term module refers to a container of components.

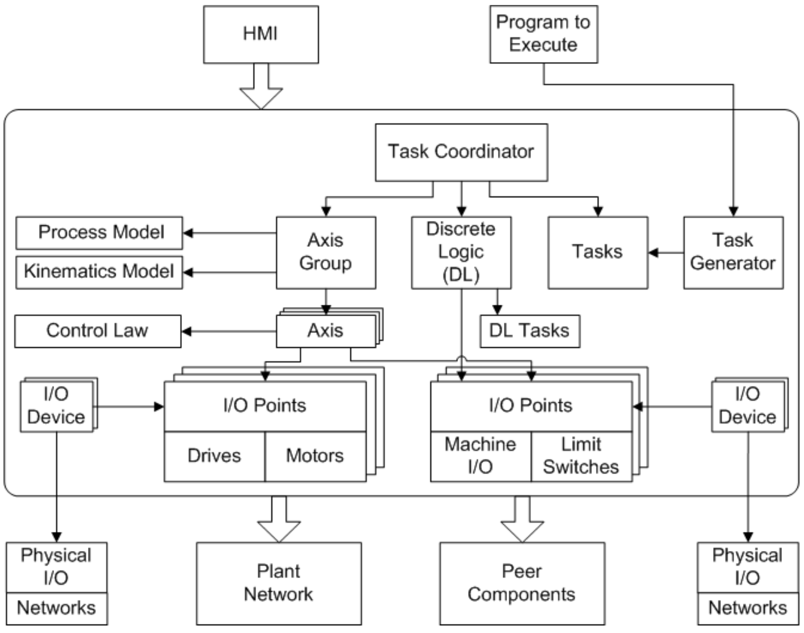

Fig. 2 OMAC architecture

Fig. 2 describes the OMAC API controller functionality. The HMI (Human Machine Interface) module is responsible for human interaction with a controller, such as presenting data, handling commands, and monitoring events. The Axis Group module coordinates the motions of individual axes, transforming an incoming motion segment specification into a sequence of equidistant time-spaced set points for the coordinated axes. The Axis Module performs servo control of axis motion, transforming incoming motion set points into set points for the corresponding actuators. The Task Coordinator module is the highest level Finite State Machine in the controller and it performs sequencing operations and coordinates the various modules in the system based on programmable Tasks. A Task Generator module generates a series of application-neutral Transient Tasks based on the application-specific control [5].

Components defined in the OMAC API specification include: IO Point, Control Plan, Kinematics, Process Model, and Control Law. Interfaces define component functionality. A component may contain multiple interfaces, either through aggregation or inheritance. New components may extend functionality by means of aggregation or specialization [6].

Two Workgroups currently operate within OMAC:

- OMAC Packaging Workgroup (OPW) has the objective to maximize the business value of packaging machinery by improving automation guidelines and standards. They work with automation suppliers, OEMs, and trade groups worldwide to encourage their support of the OMAC Packaging Guidelines throughout their products and practices, creating a mutually beneficial environment for the guidelines.
- OMAC Machine Tool Workgroup (OMW) has the objective to create an environment that maximizes the machine automation choices of end-users and OEMs and increases their flexibility through greater openness and interoperability. They work with CNC controller vendors, CAD/CAM suppliers, and CNC OEMs to encourage their support of OMW "Connect and Manufacture" standards and best practices.

2.3 OSACA

The ESPRIT project OSACA (Open System Architecture for Controls within Automation Systems) started in 1992, with the aim of uniting European interests and creating a vendor-neutral standard for open control systems. It was supported by major European control vendors [7].

The basic technical approach of the OSACA architecture is the hierarchical decomposition of control functionality into functional units. For each of these functional units (e.g. motion control, axes control or logic control), the interfaces are specified by applying object-oriented information models. The process interface consists of several process objects that are used to describe the dynamic behaviour of the application modules by means of finite state machine (FSM). It can also be used to activate specific functions in the form of local or remote procedure calls [2].

The interoperability of distributed application modules is supported by an infrastructure OSACA platform. The platform is composed of hardware and program groups (operating system, communication system) that offer a uniform service for the functional unit control [7]. The API with the functional unit (FU) is based on a well-defined task.

The three basic elements are:

- **Communication System**: hardware and software are defined independently of the interface for information exchange among different modules of the controller application. The OSACA communication system allows the information exchange between the client and server applications to be carried out in a proper way.
- **Reference Architecture** determines the control FU and specifies the external interface. This is done to enable the use and integration of external units through internal data in a well-defined way. FU examples are Man Machine Interface, Interlock Logical Control and Axis Motion Control. An external module using object-oriented communication for interfacing data with application modules is

defined for each identified FU. The interface for writing and reading data is located in the Architecture-Oriented Object and the access is available with the use of the Communication-Oriented Object.

– **System Configuration** allows a controller to generate a dynamic configuration through a combination of different application modules. This does not only allow the determination of a specific topology of a given functionality, but also the synchronization among the distributed processes.

Fig. 3 describes the platform of the OSACA system, where a configuration request generated by a microcomputer is sent to the system. The reconfiguration is based on object-oriented programs and a class library. During the reconfiguration, the functional units are used to interact with variables and internal data. The OSACA application protocol uses a client/server base mounted on the object-orientation principle. The functionality of all functional units has external access and it is configurable by the communication platform. From the customer's viewpoint, the server can be accessed through the shipping and reception of system communication messages.

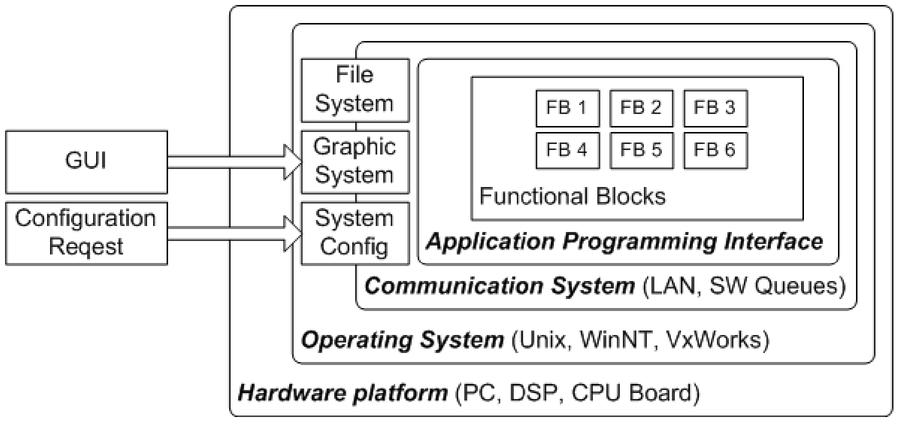
GUI
Configuration Reqest
File System
Graphic System
System Config
FB 1
FB 2
FB 3
FB 4
FB 5
FB 6
Functional Blocks
Application Programming Interface
Communication System (LAN, SW Queues)
Operating System (Unix, WinNT, VxWorks)
Hardware platform (PC, DSP, CPU Board)

Fig. 3 OSACA architecture

2.4 NIST

The NIST (National Institute of Standards and Technology) has been working with a variety of industry representatives and organizations to investigate the application of open architecture concepts to machine tool controllers [8]. The first studies on Open Architecture Controller began when the NIST proposed the use of RCS (Real-time Control System), which is an architecture model of 15 years ago [7]. The first version of EMC (Enhanced Machine Controller) was originally developed by the Intelligent Systems Division at the NIST [9]. The current EMC version (LinuxCNC) [10] is an actively developed and community-maintained software package, presenting an effort to simplify, organize and continuously extend the original software. LinuxCNC is a descendant of the original NIST EMC software, which is in the Public Domain. LinuxCNC has a lot of new functionalities such as Hardware Abstraction Layer that allows adaption to many kinds of machinery, software PLC controller, and a new trajectory planner.

2.5 JOP

The aim of JOP (Japanese Open Promotion Group) was to provide the opportunities for various companies to discuss and work together on the standardization of open controller technologies [2]. They published an API between the NC kernel and the HMI that is called PAPI (Principal API). The specification of the PAPI was defined and has become a new Japanese standard.

2.6 NGC

The Next Generation Workstation/Machine Controller (NGC) program based on the RCS reference model, co-sponsored by the National Center for Manufacturing Sciences (NCMS), the U.S. Air Force and Martin Marietta, organized industry requirements and prepared a specification for an Open System Architecture Standard (SOSAS) [11].

2.7 HOAM-CNC

The HOAM-CNC architecture (Hierarchical Open System Architecture Multi-Processor for CNC machines) covers the machine hardware, offering the advantages of having two buses, the CNC control bus, and another bus to allow the introduction of new components [7]. The following research centres have participated in this research:

- The University of Michigan, Ann Arbor, USA – which studies the open architecture controller in real-time for machine tools of high performance. They execute the implementation of several types of hardware control with net communications to study the difference in machine performance depending on the adopted architecture.
- The British Columbia University, Vancouver, Canada - They use this architecture seeking the regulating adaptive control. Modules that detect tool damage and vibration are inserted using acoustic sensors for the control execution. A primary bus is used to execute the machine control process and to monitor the tasks and a secondary bus of higher performance is used to communicate with the CNC.

3. Using OAC in a series of controllers at the Lola Institute

Open architecture benefits are used for the development of series of controllers at the Lola Institute. The main goal is to develop an OAC system that can be easily reconfigured in order to control various types of robot manipulators, machine tools, and other multi-axis systems, such as: 6-axis robot manipulator, 5-axis machining robot, 3-axis parallel kinematic milling machine, 3-axis DELTA robot or 3-axis human centrifuge.

This design philosophy is implemented on two distributed PCs using CORBA: User PC (U-PC) and System PC (S-PC). CORBA is an acronym of Common Object Request Broker Architecture, defined by the Object Management Group (OMG). It is a vendor-independent specification that has been implemented by numerous hardware and software system manufactures, creating a rich and robust framework that successfully operates across heterogeneous computing platforms [12]. The distributed architecture also offers many benefits, such as openness, dynamical extensibility, and mobility. Robot controllers can be very complex systems that have to deal with a number of tasks in real-time. In a distributed system, demanding tasks are run on different platforms, so they do not have to compete for one processor time [13].

Both PC architectures are based on the real-time Linux platform, where OROCOS (Open RObot COntrol Software) is set. OROCOS is a European project started in 2001, with the aim to develop a general-purpose, free software and modular framework for robot and machine control. Three laboratories were participating: Katholieke Universiteit Leuven from Belgium as a project contractor, Centre National de la Recherche Scientifique (CNRS) from France and Kungliga Tekniska Högskolan (KTH) from Sweden. Many other European laboratories were participating in the discussions and design. OROCOS is one of the most comprehensive systems which has a similar approach to OACs as that described in this paper.

The control architecture used for the development of Lola controllers can be configured statically or dynamically during two stages:

- Off-line architecture configuration stage (off-line ACS).
- On-line architecture reconfiguration stage (on-line ARS).

3.1 Off-line ACS

At the beginning of the off-line ACS, the architecture requirements are defined for a particular robot manipulator, machine tool or some other multi-axis system. During the off-line ACS, the system integrator can modify and configure the architecture, if changes in requirements occur, by adding or modifying instructions, real-time state machine, calibration methods, some of modules (i.e. kinematics or PID controller) or by adding new interfaces from PC to the chosen multi-axis system. At this stage, more than one configuration can be defined for the selected controlled system.

The U-PC provides application services at the off-line ACS as well as at the on-line ARS. During the off-line ACS, this application is intended for the system integrator to define the system configuration and robot-specific parameters. The system integrator, as well as the end user, can write programs in L-IRL (Lola Industrial Robot Language) or use CAD/CAM applications to generate three-dimensional models and tool paths. The application from the U-PC calculates kinematic chains of a selected multi-axis system and its configuration. The U-PC also provides data analysis, joint constraints, and graphical simulation of robot motions. In the end, the object code is generated, P or XML code, depending on the selected one. This is a case if L-IRL code has been used to program the robot motion path. If machining operations have been developed using the CAD/CAM (computer-aided design/computer-aided manufacturing) application, an appropriate G-code is generated for a specific machine tool.

L-IRL is a procedural, high-level language for the programming and control of robot manipulators or for the cooperative work of multiple robots. It is based on DIN 66312 standard for industrial robots. Programs written in the L-IRL language are compiled and then executed on the real-time robot controller. L-IRL contains all general structures such as data processing, expressions, program flow control, procedures and operations, as well as robot specific structures, geometric data types and expressions, motion statements, etc.

During the off-line ACS, the S-PC is intended for the system integrator to define an appropriate interpreter of the object code selected at the U-PC, to set sampling rates, modify real-time state machine, calibration methods, to add new functionality to existing modules, make new modules and configure interface modules to robot joints and sensor devices.

If more than one configuration is defined, the user chooses one particular configuration of the selected configuration family at the beginning of the on-line ARS. During the on-line ARS, the architecture can be dynamically reconfigured by selecting another configuration from the selected family. It is implemented in the defined state for the reconfiguration of the state machine by deactivating the system configuration file, robot specification and modules of the current configuration and by activating appropriate files and modules of the desired configuration.

3.2 On-line ARS

When the on-line ARS is entered, all components operate in real-time. The reference architecture is shown in Fig. 4. In this case, the Lola 50 6-axis robot manipulator is controlled and its virtual model is used. The real-time control logic is implemented by using one supervisory finite state machine (FSM). The control system consists of different modules that are previously constructed and verified independently of other modules at the off-line ACS.

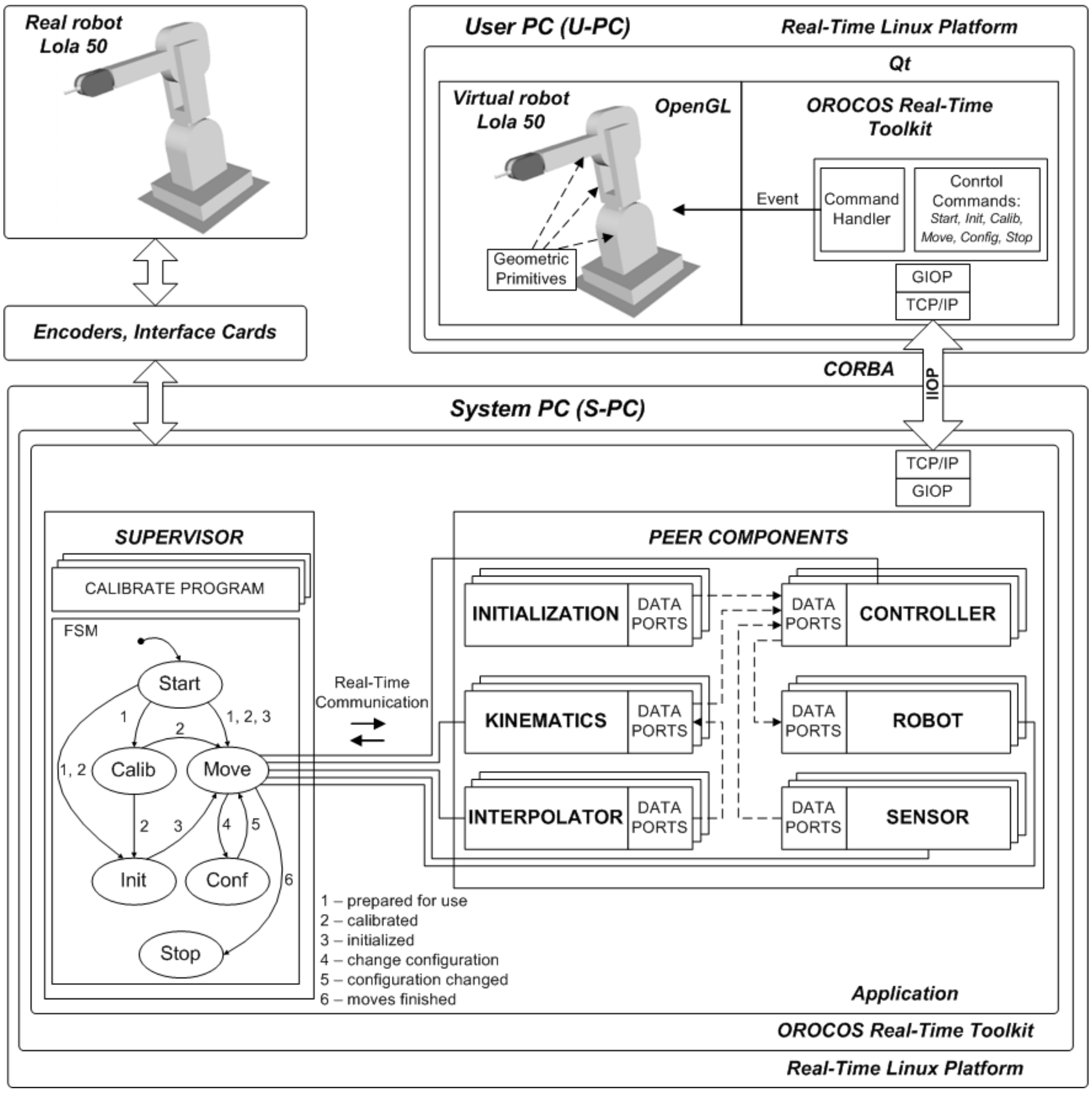

Fig. 4 OAC approach used in Lola controllers

The U-PC provides an interactive user interface. The user can personalize the user interface by including the desired parameters he would like to monitor, two or three-dimensional simulation of robot motions, information regarding the robot state, etc. The U-PC is implemented in an integrated environment using Qt, OpenGL and OROCOS. Qt libraries are used to create a GUI that includes the display of current positions, two or three-dimensional robot motions, and appropriate control buttons. OpenGL libraries are used for the drawing and linking of elementary primitives such as box, cylinder or sphere, and for the translation and rotation of robot parts in order to implement a virtual model of a real robot. OROCOS libraries are included to handle commands. The command handle is entered after a new command from S-PC is received and then the function for setting the positions of the virtual robot axes is called. As a final result, a real-time application is obtained. The application is able to accept S-PC commands, to run virtual robot functions, to display two or three-dimensional robot motions, to change the current program flow or configuration and to send commands to the S-PC [14].

The S-PC provides the loading of a chosen configuration and the execution of the object code generated on the U-PC at the off-line ACS, the control of a robot manipulator or machine tool, the acceptance of user commands for the direct control of the manipulator, the change of configuration requested from instructions in the application code or by user command, the feedback to the U-PC using CORBA in the form of robot positions, velocities, logical states, etc.

The S-PC control system is structured in layers. The bottom layer is the real-time Linux operating system whose kernel is patched with an open source real-time framework, Xenomai. The OROCOS Real-Time Toolkit (RTT) is referred to as a middleware lying between the operating system and the application level. The RTT provides the infrastructure and the functionalities to build robotics real-time applications in C++. The RTT allows the setup, distribution and building of the real-time components, which are the top layer of the control system. Besides RTT, OROCOS is composed of OCL (OROCOS Component Library), KDL (Kinematics and Dynamics Library), and BFL (Bayesian Filtering Library).

The application layer consists of tasks (components). Every component has a specific function defined at the off-line ACS (i.e. kinematics, interpolator or servo controller). Real-time state machines as well as program scripts (Calibrate Program in this case) can be integrated into the component. There is a supervisory component with a FSM loaded into it together with all the peer components defined at the off-line ACS. The peer components can be configured, started, stopped and interfaced by the FSM and can also communicate with each other through their interfaces: Properties, Events, Methods, Commands, and Data Ports. The FSM contains a collection of states whose number is defined at the off-line ACS. That way, the entire control system is represented as a single FSM at the highest hierarchical point [15].

4. Advantages of using OAC

Open control systems have a lot of advantages for control vendors, robot manipulators, machine tool builders, and end users . The main benefits are shown in Fig. 5 [16].

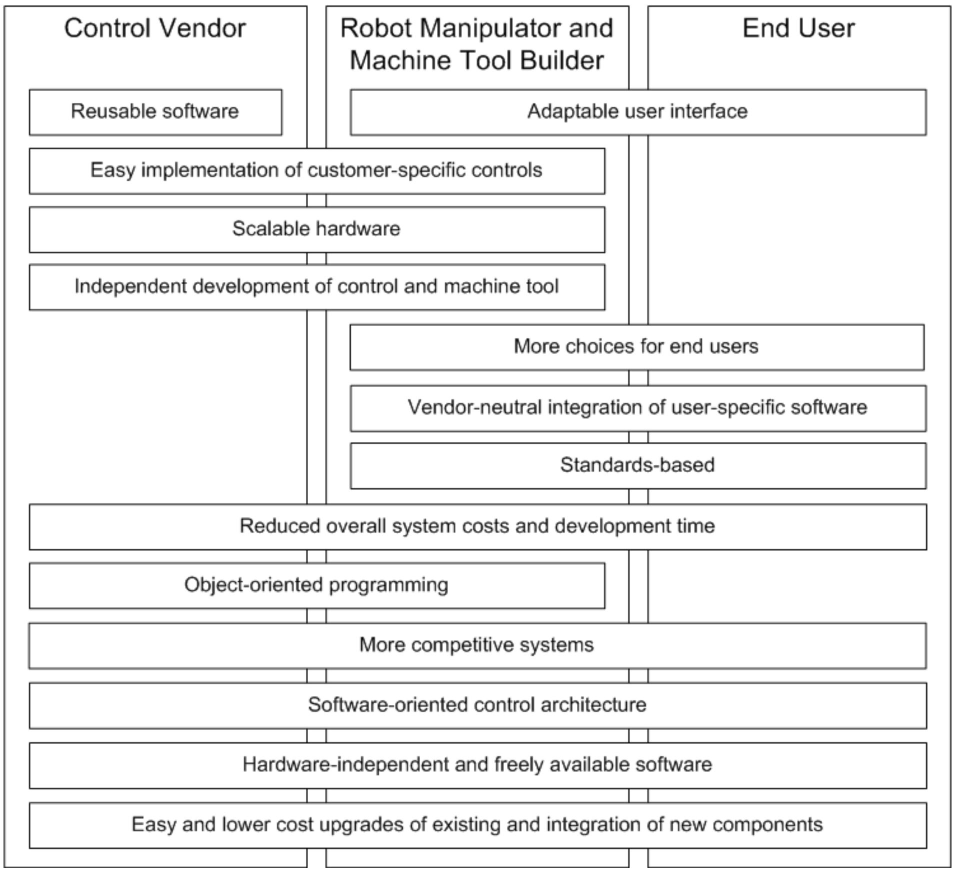
Control Vendor
Robot Manipulator and Machine Tool Builder
End User
Reusable software
Adaptable user interface
Easy implementation of customer-specific controls
Scalable hardware
Independent development of control and machine tool
More choices for end users
Vendor-neutral integration of user-specific software
Standards-based
Reduced overall system costs and development time
Object-oriented programming
More competitive systems
Software-oriented control architecture
Hardware-independent and freely available software
Easy and lower cost upgrades of existing and integration of new components

Fig. 5 Advantages of using the OAC approach

Designers of robots and multi-axis machining systems benefit from a high degree of openness covering also the internal interfaces. The external openness is much more important for users. It provides the methods and utilities for integrating the user-specific applications into the existing controls and for adapting to user-specific requirements, adaptable user interfaces or to the collection of machine and production data, for instance. The external openness is mainly based on the internal openness but has functional or performance limitations.

5. Conclusion

In this paper, various types of open architecture concepts for the control of robot manipulators, machine tools and multi-axis machining systems have been described through major international activities, (OSEC, OMAC, OSACA, NIST, JOP, NGC, and HOAM-CNC) and a series of controllers that have been developed using the OAC concept at the Lola Institute. All proposed architectures have integrated equipment of several manufacturers in order to obtain control solutions at a lower cost, while maintaining the same performance.

The short review can be summarized through the following points. Open architecture approach provides a wide range of benefits in the control of robots and multi-axis machining systems. This was the main reason for significant efforts that have been made all around the world in order to develop open architecture platforms. The resulting benefits include the software-oriented approach which leads to a reduction in the number of hardware modules, PC-based solutions with a homogenous, standardized environment, reusable software possibility, system reconfigurability, configuration and implementation of new applications and requirements, hardware changeability, adaptable interface, less time for development, and reduction in overall system costs.

At the very beginning of OAC development, the NIST proposed the use of a RCS model. The RCS model was the base to the NGC program which tried to meet the industrial needs for the next generation controllers. JOP has been designed based on the standardization of open controller technologies. In the meantime, several teams from Japan, USA and Europe have started to develop their own open architecture platforms. The OSACA architecture has been used mostly in the software area, while the OMAC architecture has been mainly active in industrial applications. The OSEC architecture has been used in automation in industrial field, logistics and support, while the HOAM-CNC architecture has been mainly oriented to the hardware area in terms of new sensors and special module implementation [7].

The basic ideas of OACs were quite similar, but the levels of abstraction were different. OMAC has been developed in more details than the others. OMAC and OSACA API properties are based on the object-oriented model and the API definition through C++ and IDL mapping to C, C++ or Java respectively, while JOP and OSEC use function calls and the C language. OMAC and OSACA provide better methods for the system configuration in comparison with NGC. On the other hand, NGC provides a full description for the interaction between components and the platform, while OMAC and OSACA do not define this interaction in detail [17]. Only OMAC is really open towards many software platforms. OSACA needs its own platform to run, so software development is needed to move to a new environment. OSEC is limited in the PC/Windows world [18].

Development of controllers at the Lola Institute based on the OAC approach has also been presented. The main aim was to develop a distributed, reconfigurable OAC system that can be easily adopted in order to control various types of robot manipulators, machine tools and other multi-axis systems. Currently, controllers for the 6-axis robot manipulators, Lola 15 and Lola 50, have been fully implemented, while the controllers for a 3-axis human centrifuge and a 4-axis spatial disorientation trainer are under development. Future research will be related to the implementation of OACs to a 3-axis parallel kinematic milling machine and a 3-axis DELTA robot.

6. Acknowledgment

Part of this research related to the Lola controllers was carried out within the research project that is supported by the Ministry of Science and Technological Development, Republic of Serbia: “Development of the devices for pilot training and dynamic flight simulation of modern combat aircraft: 3 DoF centrifuge and 4 DoF spatial disorientation trainer”.

REFERENCES

[1] D. Dumur, P. Boucher, J. Roder: *Advantages of an Open Architecture Structure for the Design of Predictive Controllers for Motor Drives*, Annals CIRP 47(1), 1997, pp. 269–274.

[2] G. Pritschow, Y. Altintas, F. Jovane, Y. Koren, M. Mitsuishi, S. Takata, et al.: *Open Controller Architecture - Past, Present and Future*, Annals of the CIRP, 2001, vol. 50, no. 2, pp. 463-470.

[3] OSE Consortium, *OSEC II, Project Report*, October 2006.

[4] *OMAC,* Available at: http://www.omac.org

[5] S. Birla, D. Faulkner, J. Michaloski, S. Sorenson, G. Weinert, J. Yen: *Reconfigurable Machine Controllers using the OMAC API*, Proceedings of the CIRP 1st International Conference on Reconfigurable Manufacturing. May 2001.

[6] J. Michaloski: *Analysis of Module Interaction in an OMAC Controller*, Proceedings of the World Automation Congress Conference (WAC 2000), Maui, HI, June 11-16, 2000.

[7] O. Asato, E. Kato, R. Inamasu, A. Porto: *Analysis of Open CNC Architecture for Machine Tools*, Journal of the Brazilian Society of Mechanical Sciences, Print version, 2002, vol. 24, no. 3, pp. 208-212.

[8] F. Proctor, et al.: *Open Architecture for Machine Control.* NISTIR-5307, Tech. rep., NIST, MD, 1993.

[9] T. Staroveški, D. Brezak, T. Udiljak, D. Majetić: *Implementation of a LINUX-Based CNC Open Control System*, 12th International Scientific Conference on Production Engineering, Computer Integrated Manufacturing and High Speed Machining, CIM, Biograd, June 2009.

[10] *LinuxCNC User Manual*, Available at: http://www.linuxcnc.org/docs/LinuxCNC_User Manual.pdf

[11] *Next Generation Controller Specification for an Open Systems Architecture Standard*, Manufacturing Technology Directorate Wright Laboratory, September 1994.

[12] Y. Gong: *CORBA Application in Real-Time Distributed Embedded Systems*, Survey Report ECE 8990 Real-Time Systems Design, Spring 2003.

[13] A. A. Loukianov, H. Kimura, M. Sugisaka: *Implementing Distributed Control System for Intelligent Mobile Robot*, Artificial Life and Robotics, 2004, vol. 8, no. 2, pp. 159-162.

[14] G. Ferenc, J. Vidaković, M. Lutovac, Z. Dimić, V. Kvrgić: *Distributed Robot Control System Implemented on the Client and Server PCs Based on the CORBA Protocol*, Mediterranean Conference on Embedded Computing 2012, pp. 158-161.

[15] G. Ferenc, M. Lutovac, J. Vidaković, Z. Dimić:V. Kvrgić: *Real-Time Robot Control Logic using Modular FSM*, International Conference Management of Technology - Step to Sustainable Production 2012, pp. 259-265.

[16] F. Proctor: *Practical Open Architecture Controllers for Manufacturing Applications*, Open Architecture Control Systems, ITIA Series, 1998.

[17] C. Yonglin: *An Evaluation Space for Open Architecture Controllers*, The International Journal of Advanced Manufacturing Technology, vol. 26, issue 4, 2005, pp. 351-358.

[18] J. Nacsa: *Comparison of Three Different Open Architecture Controllers*, Proceedings of IFAC MIM, Prague, 2001, pp. 134-138.

Submitted: 21.11.2012

Accepted: 22.02.2013

Goran Ferenc
Zoran Dimić
Maja Lutovac
Jelena Vidaković
Vladimir Kvrgić
Lola Institute
Kneza Višeslava 70a
11030 Belgrade Serbia