Open Controller Architecture – Past, Present and Future

Günter Pritschow (Co-ordinator), Yusuf Altintas, Francesco Jovane, Yoram Koren,
Mamoru Mitsuishi, Shozo Takata, Hendrik van Brussel, Manfred Weck, Kazuo Yamazaki

Abstract
Open Control Systems are the key enabler for the realization of modular and re-configurable manufacturing systems. The large number of special purpose machines and the high level of automation have led to an increasing importance of open control systems based on vendor neutral standards. This paper gives an overview on the past, present and future of Open Controller Architecture. After reflecting onthe different criteria, categories and characteristics of open controllers in general, the CNC products in the market are evaluated and an overview on the world-wide research activities in Europe, North America and Japan is given. Subsequently the efforts to harmonize the different results are described in order to establish a common world-wide standard in the future. Due to the "mix-and-match" nature of open controllers concentrated attention must be paid to testing mechanisms in the form of conformance and interoperability tests.

Keywords: Open architecture control, CNC, Machine tool

1 INTRODUCTION

Open Architecture Control (OAC) is a well known term in the field of machine control. Since the early nineties several initiatives world-wide have worked on concepts for enabling control vendors, machine tool builders and end-users to benefit more from flexible and agile production facilities. The main aim was the easy implementation and integration of customer-specific controls by means of open interfaces and configuration methods in a vendor-neutral, standardized environment [13][19].

The availability and broad acceptance of such systems result in reduced costs and increased flexibility. Software can be reused and user-specific algorithms or applications can be integrated. Users can design their controls according to a given configuration. This trend was forced both by the increasing number of special purpose machines with a high level of automation and the increasing development costs for software (Figure 1).

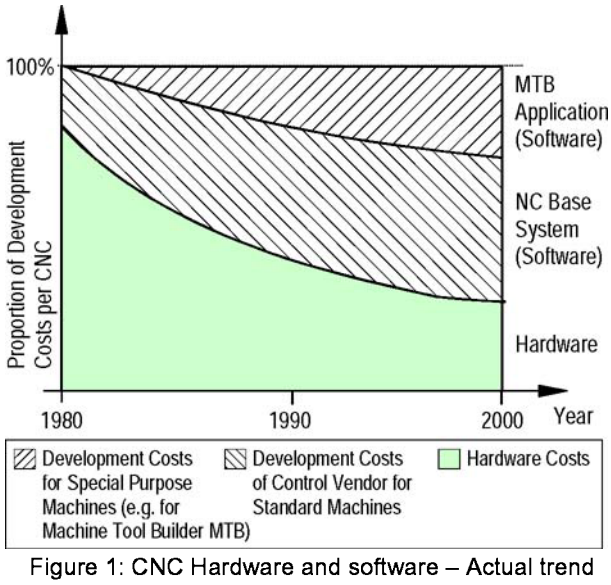

Figure 1: CNC Hardware and software – Actual trend

In the past the CNC market was dominated by heterogeneous, device-oriented systems with proprietary hardware and software components. The tight coupling of application software, system software and hardware led to very complex and inflexible systems. Great efforts were made to maintain and further develop the products according to new market requirements. Modern CNC approaches, which comprise extensive functionality to achieve a high quality and flexibility of machining results combined with a reduced processing time, favor PC-based solutions with a homogenous, standardized environment (Figure 2). The structure is software-oriented and configurable due to defined interfaces and software platforms. Open control interfaces are necessary for continuously integrating new advanced functionality into control systems and are important for creating re-configurable manufacturing units [17]. Unbundling hardware and software allows profiting 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 increases simply by upgrading the hardware platform.

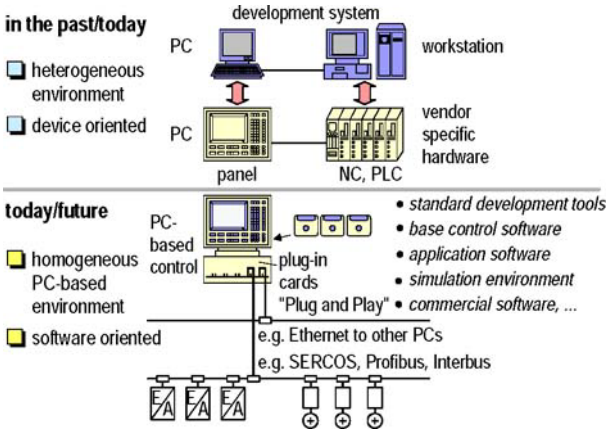

Figure 2: PC-based, software-oriented Control Systems

There are a lot of benefits for suppliers and users of open control systems (Figure 3) [7]. CNC designers and academics benefit from a high degree of openness covering also the internal interfaces of the CNC. For CNC users the external openness is much more important. It provides the methods and utilities for integrating user-specific applications into existing controls and for adapting to user-specific requirements, e.g. adaptable user interfaces or collection of machine and production data. The external openness is mainly based on the internal openness but has functional or performance limitations.

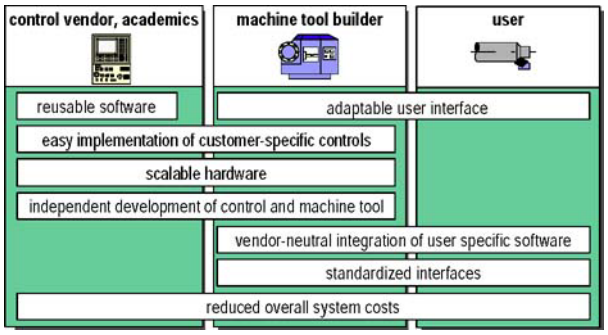

Figure 3: Benefits of using Open Control Systems

2 STATE OF THE ART

2.1 Control Systems and their interfaces

Controls are highly sophisticated systems due to very strict requirements regarding real-time and reliability. For controlling the complexity of these systems hardware and software interfaces are an essential means. The interfaces of control systems can be divided into two groups – external and internal interfaces (Figure 4).

External Interfaces

These interfaces connect the control system to superior units, to subordinate units and to the user. They can be divided into programming interfaces and communication interfaces. NC and PLC programming interfaces are harmonized by national or international standards, such as RS-274, DIN 66025 or IEC 61131-3. Communication interfaces are also strongly influenced by standards. Fieldbus systems like SERCOS, Profibus or DeviceNet are used as the interface to drives and I/Os. LAN (Local Area Network) networks mainly based on Ethernet and TCP/IP do reflect the interfaces to superior systems.

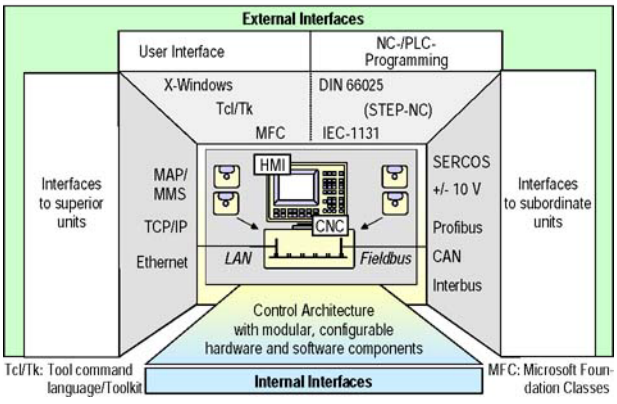

Figure 4: External and internal interfaces of control systems

Internal Interfaces

Internal interfaces are used for interaction and data-exchange between components that build up the control-system core. An important criterion in this area is the support of real-time mechanisms.

To achieve a re-configurable and adaptable control the internal architecture of the control system is based on a platform concept. The main aims are to hide the hardware-specific details from the software components and to establish a defined but flexible way of communication between the software components. An application programming interface (API) ensures these requirements. The whole functionality of a control system is subdivided into several encapsulated, modular software components interacting via the defined API.

2.2 Hardware and software structure of control systems

Figure 5 shows different variants for the hardware structures of control systems. Variant a) shows an analog drives interface with position controller in the control system core. Each module of this structure uses its own processor which leads to a large variety of vendor-specific hardware. Combining modules leads to a significant reduction of the number of processors. Variant b) shows intelligent digital drives with integrated control functionality, which result from higher capacity, miniaturization and higher performance of the processors. Variant c) shows a PC-based single processor solution with a real-time extension of the operating system. All control-functions run as software tasks in the PC-based real-time environment.

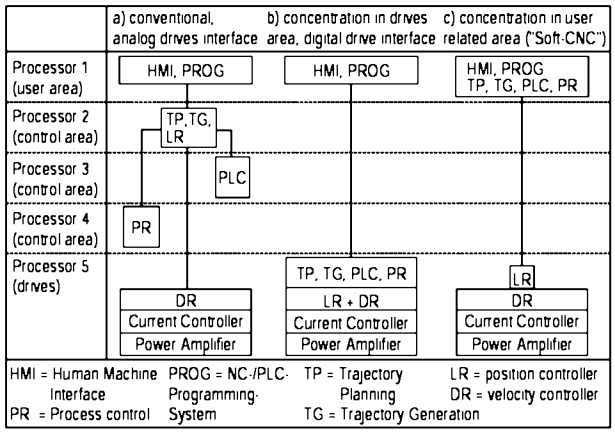

Figure 5: Hardware and software structure variants of CNCs

2.3 Market overview

The controls available in the market provide different levels of openness according to the criteria shown in Figure 6. An important criterion is the use of a standardized computing platform (i.e. hardware, operating system and middleware) as an environment to execute the HMI and CNC software. Besides this, the connectivity of the CNC to upper and lower factory levels must be guaranteed. Application Programming Interfaces (API) are used to integrate third party software in the CNC products. Although most of today's controls offer openness concerning the operator-related control functions (Human-Machine Interface, HMI) only few controls allow users to modify their low-level control algorithms to influence the machine-related control functions.

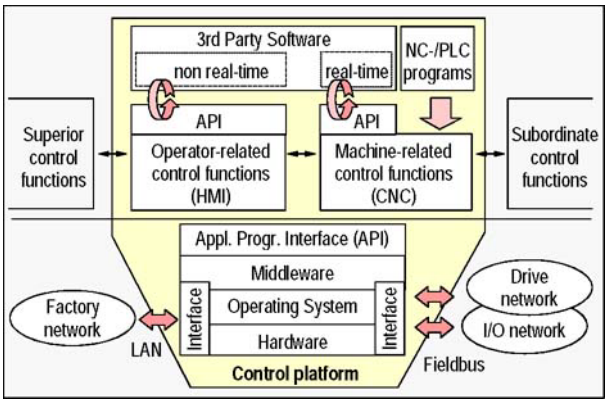

Figure 6: Criteria of Open Control Systems

Figure 7 gives an overview of the characteristics of today's control systems regarding the degree of openness.

	Criteria	Characteristics	Example
Platform	Hardware	Standard hardware with a standard bus system	PC-based control
	Operating system	Standard operating system w/o real-time	VxWorks, Windows
	Middleware	Standard middleware w/o real-time	DCOM, CORBA
Communication	Factory network	Standard physical media + communication protocol	Ethernet plus TCP/IP
	Drive network	Standard physical media + communication protocol	SERCOS interface
	I/O network	Standard physical media + communication protocol	Profibus, DeviceNet
Programming	NC/RC programming	Standard NC programming language	DIN 66025, RS-274
	PLC programming	Standard PLC programming language	IEC 61131-3
	HMI software integration	Standard application programming interface	DDE, OPC
	CNC software integration	Standard application programming interface	Compile Cycles

Figure 7: Characteristics of Open Control Systems

Although many control systems provide open interfaces for software integration (e.g. OPC) there is still no common definition of data which is passed back and forth via the programming interface. Therefore, the control systems available on the market today do not implicitly support "plug-and-play" features. To improve this situation, the fieldbus systems can serve as a role model (see Figure 8). The variety of different fieldbus systems has led to the broad consensus that harmonizing the application-oriented interfaces is desirable in order to hide the plurality and the complexity of the systems from the user. Most fieldbus organizations are already using so-called device profiles in order to support the interchangeability of the devices of different vendors.

For example, the SERCOS interface standard (IEC61491) for the cyclic and deterministic communication between CNC and drives has defined the semantics for approx. 400 parameters describing drive and control functions which are used by the devices of different vendors.

3 DEFINITIONS AND CATEGORIES OF OPENNESS

3.1 Definitions

The "Technical Committee of Open Systems" of IEEE defines an open system as follows: "An open system provides capabilities that enable properly implemented applications to run on a variety of platforms from multiple vendors, interoperate with other system applications and present a consistent style of interaction with the user" (IEEE 1003.0).

To estimate the openness of a controller the following criteria can be applied (Figure 9):

- *Portability.* Application modules (AM) can be used on different platforms without any changes, while maintaining their capabilities.
- *Extendibility.* A varying number of AM can run on a platform without any conflicts.
- *Interoperability.* AM work together in a consistent manner and can interchange data in a defined way.
- *Scalability.* Depending on the requirements of the user, functionality of the AM and performance and size of the hardware can be adapted.

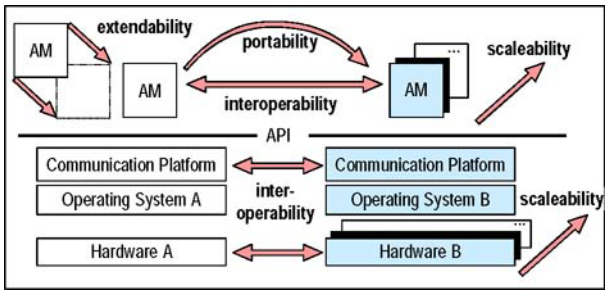

Figure 9: Criteria of Open Control Systems

	HMI	CNC	Programming Interfaces		Communication Interfaces		
CNC	**Platform (HW, OS)**	**Platform (HW, OS)**	**HMI API**	**CNC API**	**LAN Support**	**Drive Support**	**I/O Support**
Fanuc 210i/210is	PC with WinCE/NT	-- *proprietary*	++ WinAPI	-- *proprietary*	++ Ethernet	-- *proprietary*	++ Fieldbus
Indramat MTC200	PC with WinNT	O Psos	++ WinAPI	-- *proprietary*	++ Ethernet	++ SERCOS	++ Fieldbus
MDSI Open CNC	PC with WinNT+RTX (SoftCNC)		++ WinAPI	++ RT API	++ Ethernet	++ SERCOS	++ Fieldbus
Robert Bosch Typ3 osa	PC with WinNT	O PxRos	++ WinAPI	++ Job Cycles	++ Ethernet	++ SERCOS	++ Fieldbus
Allen Bradley 9/PC	PC with WinNT	-- *proprietary*	++ WinAPI	+ OCI	++ Ethernet	++ SERCOS	++ Fieldbus
Siemens E&A 840D/840Di	PC with Win95/NT	-- *proprietary*	++ WinAPI	++ Cpl. Cycles	++ Ethernet	-- *proprietary*	++ Fieldbus

RTX = Real Time Extensions for Windows ++ = Open O = Partly Open -- = Closed

Figure 8: Overview on commercial CNC systems

To fulfill the requirements of the IEEE-definition and these criteria of openness, an open control system must be:

- *vendor neutral*. This guarantees independence of single proprietary interests.
- *consensus-driven*. It is controlled by a group of vendors and users (usually in the form of a user group or an interests group).
- *standards-based*. This ensures a wide distribution in the form of standards (national/international standards or de-facto standards).
- *freely available*. It is free of charge to any interested party.

3.2 Categories of Open Control Systems

If we speak of openness in control systems, the following categories can be identified (Figure 10):

- *Open HMI:* The openness is restricted to the non-real-time part of the control system. Adaptations can be made in user oriented applications.
- *Kernel with restricted openness:* The control kernel has a fixed topology, but offers interfaces to insert user-specific filters even for real-time functions.
- *Open Control System:* The topology of the control kernel depends on the process. It offers interchangeability, scalability, portability and interoperability.

Open control systems that are available today mostly offer the possibility for modifications in the non-real-time part in a fixed software topology. They lack the necessary flexibility and are not based on vendor-neutral standards.

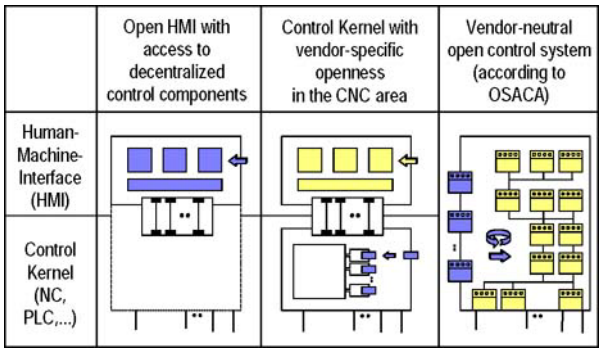

Figure 10: Levels of openness for control systems

3.3 Requirements

A vendor-neutral open control system can only be realized if the control functionality is subdivided in functional units and if well-defined interfaces between these units are specified (Figure 11). Therefore modularity can be identified as the key for an open system architecture. In determining the module complexity there is an obvious trade-off between the degree of openness and the cost of integration [6]. Smaller modules provide a higher level of openness and more options, but increase the complexity and integration costs. Furthermore such a low level of granularity can lead to much higher demands for resources and it may even deteriorate the real-time performance of the overall system.

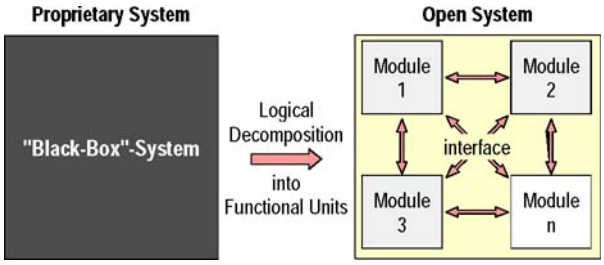

Figure 11: Decomposition of control functionality

Combining modules in the manner of "mix-and-match" requires a comprehensive set of standard Application Programming Interfaces (APIs). For vendor-neutral open control systems the interfaces need to be standardized and broadly accepted. Due to the complexity of such modular systems the definition of a system architecture is recommendable and helpful. This leads to the introduction of so-called system platforms (Figure 12). These platforms encapsulate the specifics of a computing system by absorbing the characteristics of hardware, operating system and communication. The availability of such middleware systems facilitates the easy porting of application software and also the interoperability of application modules even in distributed heterogeneous environments.

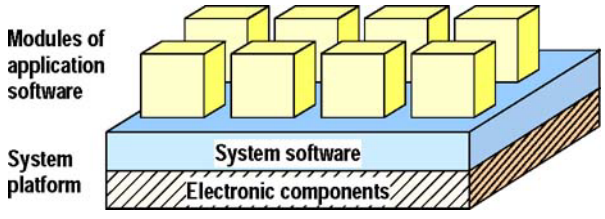

Figure 12: Infrastructure for open modular control systems

Due to the possibility to "mix-and-match" modules via standardized interfaces the quality of the overall system is determined by the degree of the interoperability between the single modules (see Section 5).

4 SYSTEMS ON THE WAY TO THE MARKET

4.1 Major international activities

OSEC (Japan)

The OSE (Open System Environment for Manufacturing) consortium was established in December 1994. Three project phases were carried out until March 1999 [1][2][3]. The OSEC Architecture was intended to provide end users, machine makers, control vendors, software vendors, system integrators, etc. a standard platform for industrial machine controllers, with which they can add their own unique values to the industrial machines, and hence promote the technical and commercial development of the industrial machines. The OSEC API 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. 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 architecture level (Figure 13). Although the structure of functional blocks can be defined uniquely by the OSEC architecture from a logical point of view, the system is neither determined nor limited at its implementation phase because there are so many options for implementations. These options may include system contrivances such as device driver, interprocess 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 to realize the architecture model is not limited to a particular model. In this way, it is assured to incorporate various ideas in the implementation model depending on the system size or its hardware implementation and/or utilization.

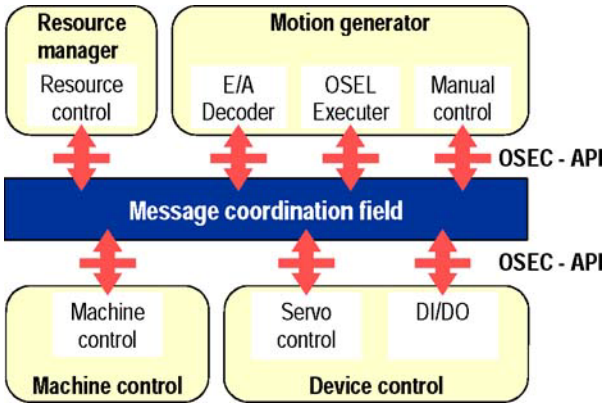

Figure 13: OSEC Architecture

JOP (Japan)

In parallel to the OSE consortium activities, MSTC formed the Open-Controller Technical Committee (OC-TC) from 1996 to 2000, under the umbrella of JOP (Japanese Open Promotion Group). The objectives of OC-TC were to provide the opportunities for various companies to discuss and work together on the standardization of open controller technologies. The OC-TC was also expected to act as liaison between domestic and international activities in this field. OC-TC was participated by approximately 50 members, which included major Japanese controller vendors, machine tool builders, integrators, users, and academics. Some of the members represented the other groups concerning open controllers such as the OSE consortium and the FA Intranet Promotion Group.

One of the working groups was engaged in developing a standard API for interfacing between NC and PC-based HMI. It should be also effective for the communication between NC and an upper level management controller. The work was carried out based on the proposals from the major controller vendors and that from the OSE consortium. The developed specifications were named PAPI and released July, 1999 [4][5]. PAPI was approved as a JIS (Japan Industrial Standard) technical report and published in October, 2000. To demonstrate the effectiveness of the specifications developed by OC-TC, in Nagoya in October 1999, two CNCs manufactured by different vendors were connected to a Windows NT machine in which the same HMI systems developed by the University of Tokyo were implemented (Figure 14).

Since any specific controller architecture is not assumed, PAPI can be implemented in various types of existing CNC systems, such as PC + proprietary NC, PC + NC board, and Software NC on PC + I/O board. The HMI system communicates with the CNCs via PAPI which is a function-oriented software library in the programming language C. The PAPI interface is neutralizing the vendor-specific interface by mapping the PAPI calls to the vendor-specific API and protocol.

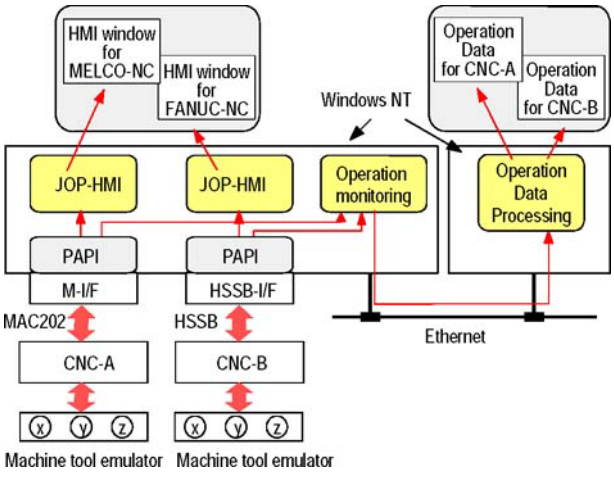

Figure 14: Demonstration system at Mechatro-tech Japan in 1999

OMAC (USA)

The Open Modular Architecture Controllers (OMAC) Users Group is an industry forum to advance the state of controller technology [10]. An effort was undertaken within OMAC to define API specification for eventual submittal to an established standards body.

The OMAC API adopted a component-based approach to achieve plug-and-play modularization, using interface classes to specify the API [11]. For distributed communication, 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 component collaboration is performed in a known manner. The term component applies to reusable pieces of software that serves as a building block within an application while the term module refers to a container of components. Tasks are components used to encapsulate programmable functional behavior consisting of a series of steps that run to completion, including support for starting, stopping, restarting, halting, and resuming, and may be run multiple times while a controller is running. Tasks can be used to build controller programs consisting of a series of Transient Tasks, with ability to restart and navigate, or as standalone Resident Tasks to handle specialized controller requirements, (e.g., axis homing or ESTOP).

To integrate components, a framework is necessary to formalize the collaborations and other life cycle aspects in which components operate. The OMAC API uses Microsoft Component Object Model (COM) as the initial framework in which to develop components, with the expected benefit that control vendors could then concentrate on application-specific improvements that define their strategic market-share – as opposed to spending valuable programming resources reinventing and maintaining software "plumbing." The primary problem with COM framework, specifically under the Windows 2000 operating system, is the lack of hard, real-time preemptive scheduling, but third party extensions to Windows 2000 can be used to overcome this requirement.

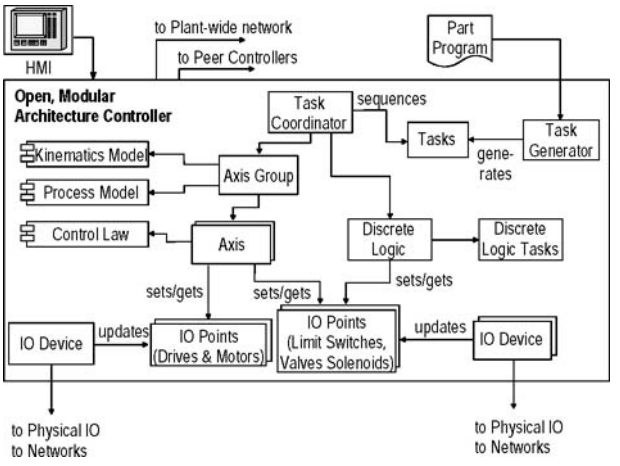

Figure 15. Sketch of Open Modular Architecture Controller API Functionality

Figure 15 illustrates a sketch of OMAC API controller functionality. The HMI module is responsible for human interaction with a controller including presenting data, handling commands, and monitoring events and in the OMAC API "mirrors" the actual controller with references to all the major modules and components via proxy agents. The Task Coordinator module is responsible for sequencing operations and coordinating the various modules in the system based on programmable Tasks. The Task Coordinator can be considered the highest level Finite State Machine in the controller. A Task Generator module translates an application-specific control program (e.g., RS 274 part program) into a series of application-neutral Transient Tasks. The Axis Group module is responsible for coordinating the motions of individual axes, transforming an incoming motion segment specification into a sequence of equi-time-spaced setpoints for the coordinated axes. The Axis module is responsible for servo control of axis motion, transforming incoming motion setpoints into setpoints for the corresponding actuators IO points. The Control Law component is responsible for servo control loop calculations to reach specified setpoints.

OSACA (Europe)

In Europe the ESPRIT project OSACA (Open System Architecture for Controls within Automation Systems) was initiated in 1992 with the aim to unite European interests and to create a vendor-neutral standard for open control systems [9][16]. It was supported by major European control vendors and machine tool builders. OSACA reached a mature state already in April 1996 having at its disposal a stable set of specifications and a tested pool for system software. Based on these results, several application-oriented projects were carried out. In 1998 two pilot demonstrators in the automative industry proved the interoperability of OSACA-compliant controllers and applications. The OSACA Association with currently 35 members from all over the world is the lasting organization to keep and maintain the OSACA-related specifications.

The basic technical approach of the OSACA architecture is the hierarchical decomposition of control functionality into so-called functional units (Figure 16). For each of these functional units (e.g. motion control, motion control manager, axes control, logic control, etc.) the interfaces are specified by applying object-oriented information models. This technique is similar to the approach of MAP/MMS but with a limited and manageable number of object classes.

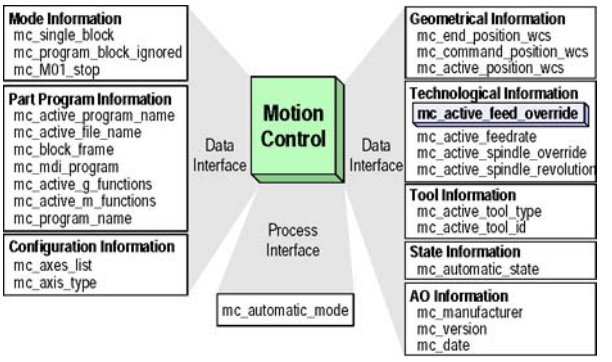

Figure 16: Example of a Data and Process Interface for the Functional Unit "Motion Control"

The data interface consists of several variable objects that support the read and/or write access to data structures (data flow). The data can be of a simple type or of a complex type (array, structure, union). By using formal templates (Figure 17) all the characteristics of a single interface object are specified. These elements cover the name (e.g. "mc_active_feed_override"), the type (e.g. UNS32: 32-bit unsigned value), the scaling (e.g. 0.1%), the range and the access rights (read only, write only, read/write) of the data. An additional description is to avoid misinterpretations of the use of the data.

The process interface consists of several process objects that are used to describe the dynamic behavior (control flow) of the application modules by means of finite state machine (FSM). The state machines are described by static states, dynamic states and transitions to change the states of a given state-machine. The transitions can handle input and output parameters to pass data between application modules via the communication platform. The formal template for such process interfaces consists of an unambiguous description and the following attributes : list of static states (identifier, list of possible transitions), list of dynamic states (identifier) and a list of transitions (input parameters, output parameters, return codes). The process interface can also be used to activate application-specific functions in form of local or remote procedure calls.

The interoperability of distributed application modules is supported by an infrastructure (so-called OSACA platform) which comprises client-server principles, synchronous and asynchronous calls and event handling via any underlying communication protocol and media (e.g. by using the TCP/IP protocol). A dedicated configuration runtime system is handling the system's startup and shutdown. Besides, it also allows an easy reconfiguration of the system.

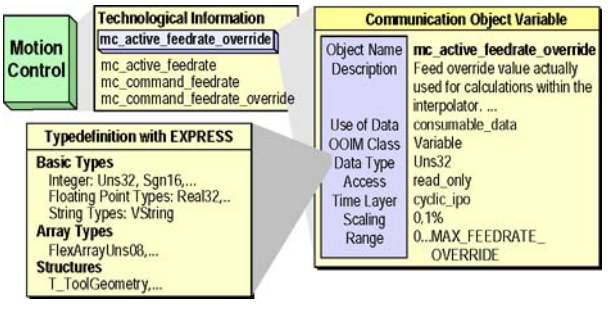

Figure 17: Use of formal templates to specify interfaces

The appliance of OSACA within several projects has proved that the formal templates (as shown in Figure 17) can not only be applied to the OSACA-specific communication objects in combination with the OSACA-specific infrastructure. Furthermore the OSACA

templates are ideally suited to standardize the data which is passed by interfaces (e.g. DDE, OPC) that only specify the access to data but not the semantics of the data itself [15].

Comparison of the different OAC approaches

The comparison of the described OAC approaches OMAC, OSACA and JOP shows that although there are common basic elements the concepts are by far not compatible to each other (Figure 18). There are major differences in the module granularity and in the type of architecture which have implications to the characteristic of the API and the modeling principles [12].

	OSACA (Europe)	JOP (Japan)	OMAC (United States)
Level of Modularity	medium	low	medium
Type of Architecture	Client-Server	Wrapper	Component-based Framework
Infrastructure	yes	no	no
Type of Appl. Prog.Interface	Object-oriented Information model	Function calls	Object-oriented Methods
Finite State Machines	yes	no	yes
Programming Language	C++	C	IDL mapping to C, C++, JAVA
Conformance Tests	yes	no	no

Figure 18: Comparison of OAC approaches

Global HMI Project

Already since 1996 the initiatives in Europe, the United States and Japan have continuously discussed the technical feasibility to harmonize the different results in order to establish a common standard in the future. After some time the groups decided to focus on the development of a global HMI standard to share a unified API for HMI design for OAC. The goal of the effort is to define a HMI API for most of the control products in the world. For this, first a common data model shall be defined. This includes naming conventions, data typing and other important attributes of the data to be described. Second, a common service model with a well-defined semantic shall be defined. Third, by combining the data and the service models with a programming and runtime environment, a common API shall be derived (Figure 19). This proceeding separates the NC-relevant technologies from the IT technology which has much shorter innovation cycles.

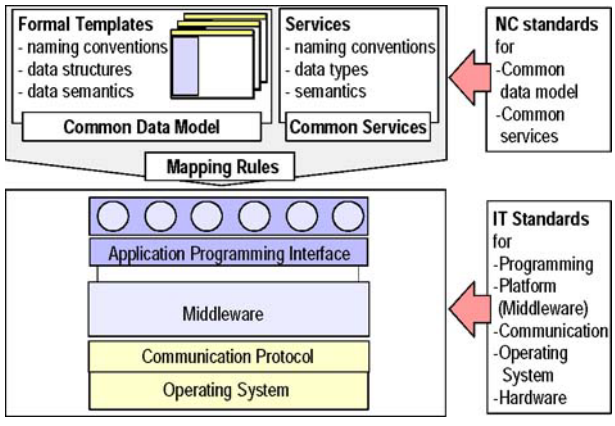

Figure 19: Common Data and Service Model for a Global HMI API

4.2 Other research activities

NIST (USA)

Part of the National Institute of Standards and Technology (NIST) mission is to develop measurements and standards for intelligent control systems pertinent to manufacturing industries. In this realm, NIST has long viewed open, modular control architectures as critical to this mission, which led to the development of the Real-time Control System (RCS) as a standard reference model for building real-time intelligent control systems [8]. RCS has evolved over many years, and has resulted in the development of numerous controller applications, including the Enhanced Machine Controller (EMC). The EMC offers real-time open-architecture control based on open source and community software development suitable for a variety of equipment, such as machine tools, robots, and co-ordinate measuring machines [20].

University of British Columbia (Canada)

The University of British Columbia has developed a user friendly, reconfigurable and modular tool kit for motion and machining process control (ORTS). The tool kit uses a script language to configure the control software in a highly open and modular way. The system can be used as an open architecture, modular operating system for the progressive development of real-time signal processing, motion and process control application. Sample applications for machine tool control and sensor assisted machining applications include CNC system for a three axis machine tool, six axis robot, piezo tool actuator, adaptive force control and tool condition monitoring [14].

University of Michigan (USA)

The University of Michigan has a long tradition in designing and developing control systems for multi-axes machines. Research activities include finite state machines (FSM) based design of machine control, implementation of supervisory control and development of Windows-based human machine interface (HMI). Actual research activities cover the design of appropriate control structures for reconfigurable machines and a common HMI API for different CNC systems.

WZL, RWTH Aachen (Germany)

At WZL in Aachen a numerical control system for 5-axis HSC-milling has been developed which is completely based on the OSACA specification. It uses the OSACA communication mechanisms for data exchange between different control modules which are distributed on a PC operated with Windows NT and a VME-System operated with VxWorks/Tornado. Further on, the conformance to the OSACA/HÜMNOS specification has been assured by following the conventions specified in the OSACA reference architecture.

ISW, University of Stuttgart (Germany)

ISW in Stuttgart has a long tradition in designing open modular control systems. In the 1980s the MPST project dealt with a modular control architecture based on hardware components integrated via a parallel bus. Later, ISW has continuously worked on a modular, software-based control architecture which is promoted and maintained by a commercial spin-off company of ISW. The Soft-CNC is based on OSACA principles and is used by many well-known European control and machine tool suppliers that want to have a maximum flexibility when realizing their automation solution. Typical applications are multi-axis machines for milling and turning, parallel kinematics machines and special-purpose machines, such as EDM and textile machines.

Collaborative research

Joint research work has been carried out between the Universities of Michigan, British Columbia, Aachen and Stuttgart. The research institutes have interfaced their specific research results to fit into the OSACA environment. UBC, ISW and WZL have cooperated to realize a gateway which allows access to the process control and monitoring tasks of ORTS via OSACA commands. The University of Michigan and ISW have jointly implemented a common HMI API for different commercial and prototype CNC systems based on OSACA principles.

5 CONFORMANCE TESTING AND CERTIFICATION

Despite the numerous and obvious advantages, vendor-neutral OAC carries some unresolved legal and technical issues. These include legal liability, loss of production during installation of new functions and ensuring technical performance after modifications and because of difficulties in honoring timing constraints and correct integration of modules [6].

In order to guarantee a maximum reliability of open controllers suppliers must offer products which meet the requirements of the given specifications and the conformity must be proved and certified by an independent organization. In conformance testing, a product is tested using specified test cases to check that it does not violate any of the specified requirements and to check that it behaves consistently with respect to the options that it is said to support. In an interoperability test, a product is tested in a multi-vendor environment. First, the product must be able to communicate with all other products in the test-bed. Second, the product must not disturb the communication between the other products. This procedure corresponds to most of the tests which are carried out by the different field-bus organizations in the market.

Over the last years, the OSACA initiative in Europe has realized a test system to check the conformance of OSACA-based implementations (client and server applications, as well as system platforms) [18]. In order to test a server implementation a reference platform and a so-called reference client is required (Figure 20). The reference client is using test scripts which describe the test and which can be executed as a batch job. The automatic test guarantees the reproducibility and the efficiency of the test even if the test itself is very extensive.

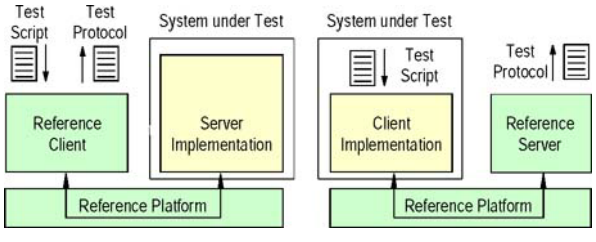

Figure 20: Conformance Test of OSACA Implementations

6 OUTLOOK AND SUMMARY

The described international initiatives and the different activities in several research laboratories world-wide show that there are many promising approaches on OAC systems. Nevertheless, it must clearly been stated that there is the risk to have many "vendor neutral" incompatible control systems instead of many "vendor proprietary" incompatible control systems. Therefore a common initiative which is supported by all major groups and the market leaders in the control business is required. To be successful in this endeavor it is important to keep the standardization of an appropriate open controller architecture independent from the mainstream development of IT standards. The Global HMI approach is a very encouraging initiative to establish a common API by defining a common data and service model. By mapping these common models to different programming and runtime environments, interchangeability and "plug-and-play" mechanisms can be achieved in reality.

7 REFERENCES

[1] OSE Consortium, OSEC-I Project Report, Sept. 1995.

[2] OSE Consortium, OSEC-II Project Report, Aug. 1996.

[3] S. Fujita, T. Yoshida, 1996, OSE: Open System Environment for Controller, 7th International Machine Tool Engineers Conference, Nov. 16-17: 234-243.

[4] Ueno, S., Chino, S., Hoshino, Y., Uneme, M., 2000, Development of the Standard Application Program Interface (API) for Open FA controller in Japan, Proc. of the 15th Annual Meeting, ASPE: 296-299.

[5] CNC Application Programming Interface, PAPI Specification 1.01E, July 26, 1999, http://www.mstc.or.jp/jop/oc/spec-e.html.

[6] Y. Koren, 1998, "Open-Architecture Controllers for Manufacturing Systems", in: "Open Architecture Control Systems", ITIA Series.

[7] F. Proctor, 1998, "Practical Open Architecture Controllers for Manufacturing Applications", in: "Open Architecture Control Systems", ITIA Series.

[8] J.S. Albus, 1991, "Outline for a Theory of Intelligence," IEEE Trans. on Systems, Man, and Cybernetics, Vol. 21, No. 3, May/June.

[9] M. Weck, 1993, "Offene NC-Systeme, Grundlage herstellerunabhängiger Flexibilität; VDI-Z 135 Nr. 5.

[10] OMAC Users Group, http://www.arcweb.com/omac.

[11] OMAC API Workgroup, "OMAC API Reference Documentation", http://www.isd.mel.nist.gov/projects/omacapi/

[12] P. Lutz, 1998, "Comparison between the OSACA and OMAC API approaches on an Open Controller Architecture", in: "Open Architecture Control Systems", ITIA Series.

[13] F. Jovane, 1998, "Open Architecture Control Systems – Summary of Global Activity", ITIA Series.

[14] N.A. Erol, Y. Altintas, M.R. Ito, "Open System Architecture Modular Tool Kit for Motion and Machining Process Control", ASME/IEEEE J. Mechatronics, vol. 5, no.3: 281-291.

[15] G. Pritschow, P. Lutz, 2000. „Design of a Common Data Model for Vendor-independent Open Control Systems", WAC'2000 Conference.

[16] G. Pritschow, G. Junghans, 1993, "Open System Controllers - a Challenge for the Future of the Machine Tool Industry", Annals of CIRP, 42/1: 449-452.

[17] Y. Koren et.al., 1999, "Reconfigurable Manufacturing Systems", CIRP Annals, vol. 2: 527-540.

[18] K. Wälde, 1999, "Certification Tool for OSACA Software", 10th DAAM Symposium, Vienna.

[19] D. Binder, 1996, "Wie offen hätten Sie's denn gern ?" – Offene Systeme in der Fertigung, Aachener Werkzeugmaschinenkolloquium.

[20] F. Proctor, "The Enhanced Machine Controller," http://www.isd.mel.nist.gov/projects/emc/emc.htm