An Open Architecture Real-Time Controller for Machining Processes

Jaehyun Park [*], Zbigniew J. Pasek [†], Yansong Shan [‡],
Yoram Koren [°], Kang G. Shin [•], and Galip Ulsoy [§]

The University of Michigan, Ann Arbor, MI 48109

Abstract

This paper presents an open architecture controller (OAC) for advanced machining and describes the OAC testbed at the University of Michigan. Because our OAC is designed for fully open systems, it does not depend on specific hardware or software components. This openness includes software resuability which enables integration of a wide range of monitoring and control features. Besides openness, our OAC system provides guaranteed real-time operation, an important requirement for advanced manufacturing.

1 Introduction

To develop a next-generation manufacturing system with flexibility, while minimizing life-cycle cost for a machine controller, as well as for developing the control system itself, it is necessary to define hardware and software architectures based on an open architecture concept. In this paper, we present an Open Architecture Controller (OAC) for machining systems and the OAC testbed we have been building up at the University of Michigan.

Previous OAC approaches could be classified into two major groups: one driven by industry focusing on the compatibility among commercial products, and the other offering hardware flexibility and software adaptability driven by the need of basic research organizations. Although these two approaches are different, the final goal of developing an OAC is to provide an *open system* for manufacturing.

The OSACA (Open System Architecture for Controls within Automation systems; ESPRIT III project 6379) project may be one of the largest-scale projects for OAC [5, 6], in which almost all of standardization matters including networking, application software as well as hardware, have been considered. The National Institute of Standards and Technology (NIST) proposed and used the RCS (Real-time Control System) reference model architecture over the past 15 years. Based on the RCS reference model, the National Center for Manufacturing Sciences (NCMS) and the U.S. Air Force co-sponsored the Next Generation Controller Program (NGC), and Martin Marietta organized industry requirements [4] and prepared a specification for an open systems architecture standard (SOSAS) [3]. The next step beyond NGC/SOSAS by NIST is the Enhanced Machine Controller Architecture (ECA) project [7, 8]. In the ECA project, an open machine tool has been implemented based on the NGC/SOSAS and RCS reference model. Other research projects like

[*] Research fellow, Dept. of EECS
[†] Research fellow, Dept. of MEAM
[‡] Research fellow, Dept. of MEAM
[°] Professor, Dept. of MEAM
[•] Professor, Dept. of EECS
[§] Professor, Dept. of MEAM

the Chimera project at Carnegie Mellon University [10], the Multiprocessor Database Architecture for Real-Time Systems (MDARTS) at the University of Michigan [2], and the Hierarchical Open Architecture Multi-Processor Motion Control System (HOAM-CNC) at the University of British Columbia [1], have demonstrated a variety of approaches to the OAC.

Although it is very difficult to form a universal agreement on the definition of an open system, basically, *vendor-neutrality* and *component-integration* are thought of as two fundamental features of an open system. Vendor-neutrality represents the requirement "an open system should be designed based on well-established standards that are independent of a single proprietary vendor". The component-integration feature represents the requirement that an open system should be highly portable and expandable incrementally. Although these two basic requirements could be accepted widely, each OAC project usually defines its own specific constraints and requirements such as *portability, interoperability, scalability, interchangeability, modularity, extensibility, reusability*, and *compatibility.*

With respect to application programs, this OAC definition enables us to integrate any new requirements for machine control in a modular manner. For example, if temperature compensation is needed, the system should be able to integrate existing results from thermal compensation research (i.e., temperature sensors, thermal models, compensation algorithms) into the controller with minimal effort. Hence, an open system should have compatibility and interoperability for vendor-neutral openness, and modularity and reusability for component-integration.

In addition to the openness requirements, the OAC must provide guaranteed real-time performance which is one of the fundamental features of automated manufacturing systems. A control task in a manufacturing system consists of several sub-tasks, such as sensing the machine status, several levels of control algorithms, and controlling actuators. Some of these tasks are executed periodically while others aperiodically. However, all of them must meet certain timing constraints. In a real-time system like a manufacturing system, monitoring/control becomes meaningless if these time constraints are not met. However, the issue of meeting real-time constraints has not been adequately addressed in previous research on OAC. Thus, to develop an advanced manufacturing controller, we must achieve two goals. The first goal is to build a flexible open system to meet the need of integrating advanced machine monitoring and control technologies in a modular manner. The second goal is to build a system with guaranteed task response times at different levels of hierarchy and real-time interfaces between a machine tool application task components in an advanced manufacturing system.

In this paper, we describe our efforts at the University of Michigan in building an open architecture real-time controller for manufacturing systems, or UMOAC (the University of Michigan Open Architecture Controller) for short, that meets the above requirements. Section 2 describes the hardware configuration of UMOAC, and Section 3 discusses its software configuration. Section 4 introduces our laboratory evaluation system (UMOAC-testbed).

2 Hardware Configuration

The UMOAC (University of Michigan Open Architecture Controller) is designed based on two basic concepts: *openness* to meet the need of integrating advanced machine monitoring and controls in a modular manner, and *real-time operation* to guarantee task response times at different levels of the hierarchy in an advanced manufacturing system.

The base hardware configuration of UMOAC is a highly distributed system in which processing nodes are connected through a real-time link/bus. This distributed system enables us to use a range of hardware configurations. Anything from a small micro-controller to a medium-size computer can be a processing node in a particular configuration. However, regardless their size and functionality, they operate within a unified software hierarchy and maintain communication compatibility. To build a heterogeneous configuration while preserving vendor-neutrality, no specific hardware platform is defined for the UMOAC. However, each processing node adopts an industry standard architecture and components such as the VMEbus platform. Figure 1 shows a typical example of the UMOAC configuration. There are three kinds of processing nodes in this configuration: operator node, real-time computing node, and real-time control node. The operator node is usually used for non-real-time tasks such as programming and non-real-time plant monitoring. The real-time computing node deals with real-time control and monitoring such as real-time data-logging, diagnosis and scheduling. The real-time control node performs fine-grain real-time tasks including servo-level control and data acquisition.

In a distributed system like the UMOAC, the communication channel between processing nodes plays an important role for real-time performance as well as its openness. Although there are several communication protocols used for manufacturing automation (e.g., Mini-MAP, and Ethernet), to send periodic, sporadic, and non-real-time messages over a single network in a timely manner, the UMOAC adopts the CAN (Controller

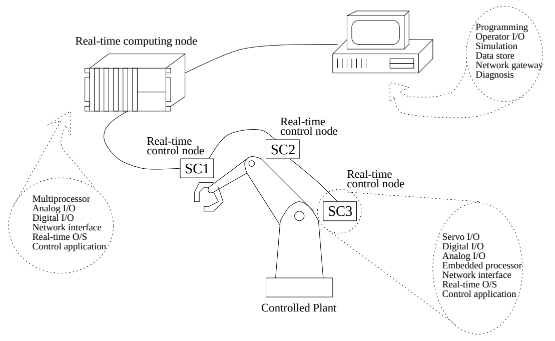

Figure 1: Hardware configuration

Area Network) [9] as a real-time communication link between processing nodes. Because it provides a small worst-case bus access latency and a distributed bus acquisition scheme based on the priority of messages, when used with a proper scheduling policy, it provides better performance in meeting real-time requirements than other existing communication protocols. We proposed a MTS (Mixed Traffic Scheduler) [12] to support periodic, sporadic, and non-real-time messages over a single CAN. Our simulation of real machine-control tasks has shown MTS to outperform DM (Deadline Monotonic) in handling high-speed real-time data.

3 Software Configuration

To enable us to write portable application programs, which should be completely isolated from the hardware configuration, the software hierarchy of UMOAC consists of three major layers: (1) application software layer, (2) object management layer, and (3) device driver layer as shown in Figure 2.

The application layer is composed of application programs, functional modules, and abstract machine models. Application programs are top-level software which includes the user interface, programming, and monitoring. To make this application program portable, abstract machine models and highly

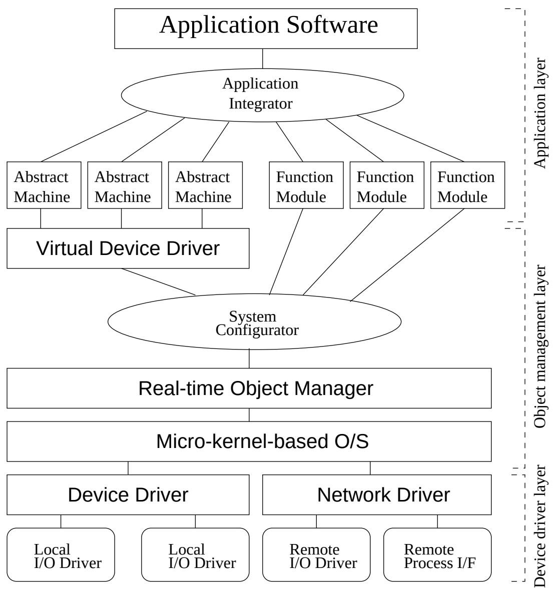

Figure 2: UMOAC Software hierarchy

modular functional modules are used, which are independent of hardware configuration. The functional modules and abstract machine models are managed by an application integrator, with which functional modules written for a specific application program can be reused for other applications and can also be extended easily.

An abstract machine model is an abstract definition which corresponds to a real machine's hardware. This abstract machine model contains a specification of the machine itself as well as the data acquired at run-time. The run-time data is managed by the real-time object manager while ensuring the predefined response time. Since an application program as well as functional modules interact only with the abstract machine models, they are isolated from hardware. This isolation enables the modules to maintain modularity and reusability.

The second software layer of the UMOAC, the object management layer, consists of virtual device driver, system configurator, real-time object manger, and real-time operating system. The main role of the system configurator is a mapping between hardware-independent application software (including functional modules and abstract machine models) and real hardware such as controlled plants and remote processing modules. If the controlled plant is connected to the local I/O interface hardware, the virtual device driver is used, of which a hardware-specific device driver would eventually have an inherent interface scheme. This virtual device driver concept used in the UMOAC provides interoperability at the hardware level. If the controlled plant is connected to remote processing modules or any functional module wanting to use the data from the remote processing module, the system configurator uses a network driver for that data. Because these mappings by the system configurator are also isolated from application programs, they maximize software modularity and reusability.

The real-time object manager provides system services tuned to the domain of object-oriented machine control applications. These services extend the micro-kernel operating system services and include domain-specific scheduling of tasks and resources. The object manager also supports persistence and configuration definition. The real-time object manger is designed on top of a commercially available real-time operating system which has a micro-kernel architecture and a POSIX-compliant interface.

The third software layer of the UMOAC, the device driver layer, is the only hardware-dependent part, which is the hardware-specific implementation of virtual device driver in object management layer. Because both local and remote I/O drivers provide the same interface protocol, remote data through CAN can be handled like local data.

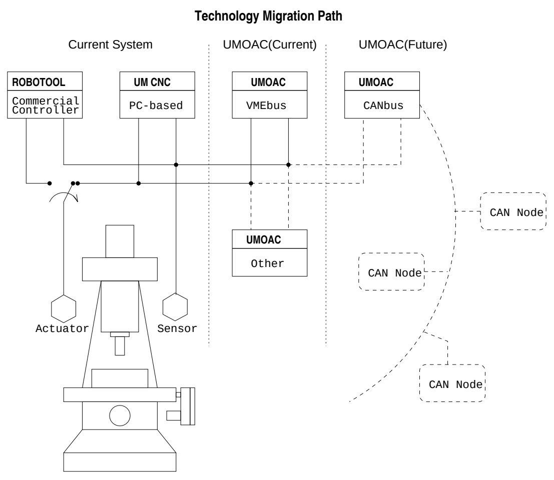

Figure 3: Hardware configuration of testbed

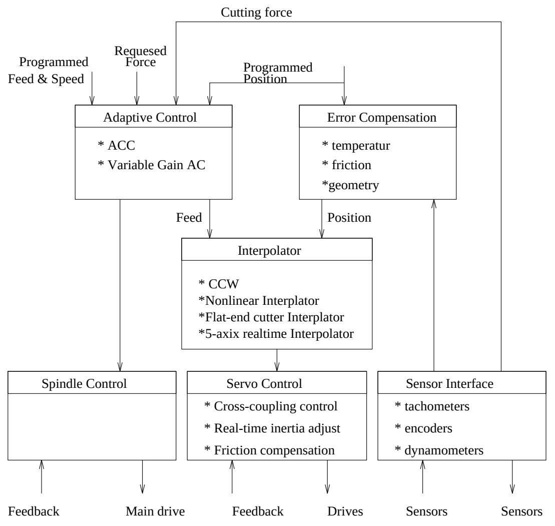

Figure 4: Testbed software configuration

4 Evaluation

An important component of our OAC research is the development of an experimental testbed in which we can implement and test openness issues. While multi-axis milling was chosen as an example application, similar approaches would be useful to other applications like robotics.

The current testbed has evolved from the activities in the University of Michigan CNC Laboratory, where our research on the next-generation CNC controllers has been conducted [11]. To effectively perform research in that area, an open and readily modifiable control system was needed — these features were not possessed by any of the commercially available CNC systems. The experimental system developed at the University of Michigan, shown in Figure 3, consists of:

- a 6-axis CNC milling machine
- an Intel i486/33MHz computer
- multiple sensors
- multiple sensor interfaces
- commercial CNC controller (Robotool).

The system was more open as compared to commercial CNC controllers, allowing researchers to implement, modify, and then test various algorithms for servo control, interpolation, adaptive control and error compensation. The system also allows the users to access data from 16 different sensors, including tachometers, digital encoders (rotary and linear), motor current sensors and a spindle power sensor. All system software was written in C. When a new control or compensation algorithm was to be investigated, the user had to code it in C in a procedure form and then combine it with the rest of existing source code necessary to run the machine. The system can also be configured according to the current needs by providing access to existing sensors.

While providing the openness necessary for research, the experimental system in Figure 4 exhibited a number of drawbacks. Its performance depends on the programmer's skill; for example, execution times of the subroutines are a function of the length of the code. Therefore, execution of critical real-time tasks cannot be strictly enforced. This issue becomes even more important as the computational load increases due to a grow-

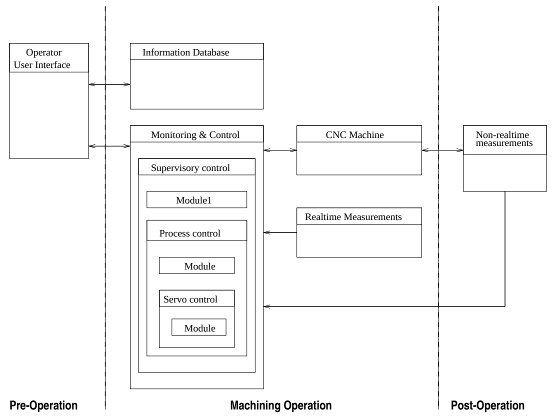

Figure 5: Testbed software configuration

ing number of involved control routines and their complexity. Also, change of the control algorithm was impossible unless the whole source code was recompiled. Moreover, the controller based on a single processor, relying on interrupts, practically excluded applications that require multitasking.

Functionally, an OAC for multi-axis machining should provide the multi-level, hierarchical configuration presented in Figure 5. The operator is provided with an efficient interface to create and update database information as well as monitoring and control routines. The operator may also acquire information from the database and control routines for diagnostic purposes. The information database and the monitoring and control routines are divided into specific modules, providing data integrity. The system's modularity simplifies installation, maintenance and upgradability of software. Monitoring and control modules have access to the information database, real-time measurements, and near real-time measurements. The information database may also be dynamically updated.

An OAC system to support process and supervisory control modules implies a need to meet multiple requirements. One important aspect from the standpoint of real-time operations is the timing constraints. Table 1

Table 1: Machining requirements

Feature	Sampling	Function	Measurements
Chatter	0.01 - 1 msec	estimate chatter avoid chatter suppress chatter	depth-of-cut spindle speed acoustic emis. cutting force feed vibrations
Cutting forces	0.01 - 1 msec	estimate cutting forces and maintain specs	depth-of-cut spindle speed acoustic emis. cutting force feed vibrations
Chip/Burr formation	10msec - 1 sec	estimate formation and maintain specs	feed spindle speed cutting force tool wear depth-of-cut
Cutting temperature	1 sec -	estimate temperature and maintain spec	feed tool wear infrared image chip contact length
Tool wear	1 msec -	estimate, maintain specified rate, compensate	depth-of-cut feed cutting force acoustic emis. surface finish part dimensions tool geometry tool vibrations spindle power
Tool failure	1 μsec - 1 msec	estimate, avoid, and detect	acoutic emission cutting force spindle power tool vibrations
Surface errors	1 sec -	estimate surface finish and tolerances, maintain specs	cutting forces tool deflections tool wear feed cutting speed cutting temperature

summarizes typical machining requirements.

The main issue in developing controllers for machining processes is the relative complexity of the system. While research on control components involved in such applications, such as servo or process controllers, is well developed, most of the research results are not implementable, due mainly to the lack of attention paid to the interactions between the various processes. The higher-level, hierarchical, supervisory control is needed to integrate and coordinate control modules. The structure of such a controller is shown in Figure 5.

The servo-level control refers here to the dynamics, hardware and controls of the basic motions of the machine tool and its auxiliary equipment, such as, for example, motor velocity and position control. The process level includes dynamics, hardware and controls involving interactions between the machine tool and the workpiece. Finally, the supervisory level corresponds to the control architecture capable of intelligent integration and coordination of involved modules at all levels by intelligently selecting appropriate process control strategies.

Supervisory control is a promising structure for implementing multiple process control strategies. It has been implemented in various forms; most of these controllers, however, were constructed in an *ad hoc* manner to solve a single problem since no rigorous controller design methodology exists.

For example, a simple face milling operation involves a number of process issues, such as occurrence of chatter, tool wear, surface finish, etc. To obtain satisfactory quality of the workpiece the following process control modules have to be used: the chatter module detects the onset of chatter and adjusts the operating point in the process input space to assure cutting process stability. The force control module manipulates the feed-rate to maximize the productivity given tool wear rate constraints. The tool wear rate constraint module provides the force controller with the current tool wear rate constraint values. The surface finish controller adjusts the feed-rate to maximize the surface finish quality. The supervisory controller integrates and coordinates the control modules to complete the operation. If an unstable depth-of-cut is attempted during the roughing pass, the force and chatter modules have to be coordinated, since their actions may contradict each other. Similarly, the actions of the surface finish module and force controller have to be coordinated during the finishing pass.

Additionally, a number of machine and process constraint modules have to be integrated within the supervisory control. Also, a database structure is needed for maintenance of machine data, model data, heuristic rules, and systematic tracking of discrete events.

5 Conclusion

Although this project is still in a preliminary stage, our approach to an open architecture control of machining systems has already had some major impact. First, our system is fully open, because it does not depend on any specific hardware or software component. Second, our system provides guaranteed real-time operation, an important requirement for advanced manufacturing. Third, our system can integrate a wide range of monitoring or control features in a modular manner. We will fully test this system on a 6-axis milling machine, and other machines at The University of Michigan.

Acknowledgments

The authors would like to thank L. Zhou, S. Jee, and R. Landers for their assistance in building our evaluation system.

References

[1] Y. Altintas and W. K. Munasinghe, "A hierarchical open-architecture CNC system for machine tools," *Annals of the CIRP*, vol. 43, no. 1, pp. 349–354, 1994.

[2] V. B. Lortz and K. G. Shin, "MDARTS: A multiprocessor database architecture for real-time systems CSE-TR-155-93," Technical report, The University of Michigan, EECS, Ann Arbor MI 48109-2122, March 1993.

[3] *Next Generation Controller Specification for an Open Systems Architecture Standard*, Manufacturing Technology Directorate Wright Laboratory, September 1994. Wl-TR-94-8033.

[4] *Next Generation Workstation/Machine Controller (NGC) Requirements Definition Document (RDD)*, 1989.

[5] G. Pritschow, "Automation technology - on the way to an open system architecture," *Robotics & Computer-Integrated Manufacturing*, vol. 7, no. 1/2, pp. 103–111, 1990.

[6] G. Pritschow, C. Daniel, G. Junghans, and W. Sperling, "Open system controllers - a challenge for the future of the machine tool industry," *Annals of the CIRP*, vol. 42, no. 1, pp. 449–452, 1993.

[7] F. M. Proctor, B. Damazo, C. Yang, and S. Frechette, "Open architecture for machine control. NISTIR-5307," Technical report, National Institute of Standards and Technology, Gaithersburg, MD 20899, December 1993.

[8] F. M. Proctor and J. Michaloski, "Enhanced machine controller architecture overview. NISTIR-5331," Technical report, National Institute of Standards and Technology, Gaithersburg, MD 20899, December 1993.

[9] *CAN Specification Version 2.0*, Robert Bosch GmbH, 1991.

[10] D. B. Stewart, R. A. Volpe, and P. K. Khosla, "Design of dynamically reconfigurable real-time software using port-based objects. CMU-RI-TR-93-11," Technical report, Carnegie Mellon University, Pittsburgh, PA 15213, July 1993.

[11] A. G. Ulsoy and Y. Koren, "Control of machining processes," *Journal of Dynamic Systems, Measurement and Control*, vol. 115, no. 2, pp. 301–308, June 1993.

[12] K. M. Zuberi and K. G. Shin, "Non-preeemptive scheduling of messages on controller area networks for real-time control applications," in *Proc. 1995 IEEE Real-Time Technology and Applications Symp.*, Chicago, U.S.A, 1995. in press.