Proceedings of the
American Control Conference
Seattle, Washington • June 1995

An Open Architecture Testbed for Real-Time Monitoring and Control of Machining Processes

Jaehyun Park, Sushil Birla,
Kang G. Shin,

Dept. of Electrical Engineering
and Computer Science
The University of Michigan
{jaehyun, birlas, kgshin}@eecs.umich.edu

Zbigniew J. Pasek, Galip Ulsoy,
Yansong Shan, Yoram Koren

Dept. of Mechanical Engineering
and Applied Mechanics
The University of Michigan
{zbigniew, ulsoy, ykoren}@engin.umich.edu

Abstract

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

1. Introduction

To develop a next-generation manufacturing system with flexibility, while minimizing life-cycle cost for the 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 OAC (Open Architecture Controller) for machining systems and the OAC testbed we have been building up at the University of Michigan.

The OAC approaches known to date 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 which are driven by the needs of basic research organizations. Although they differ in approach, 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 [1, 2] may be one of the largest-scale projects for OAC, in which almost all of the standardization matters including network, application software as well as hardware, have been considered. 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 [3] and prepared a Specification for an Open Systems Architecture Standard (SOSAS) [4]. The next step beyond NGC/SOSAS by NIST is the ECA (Enhanced Machine Controller Architecture) project [5, 6]. Other research projects like as the Chimera project [7], the MDARTS [8], and the HOAM-CNC [9] have demonstrated a variety of approaches to OAC.

Although it is very difficult to form a universal agreement on the definition of an open system, (1)*vendor-neutrality*, and (2)*component-integrability* can be thought of as two fundamental features of an open system. Vendor-neutrality is necessary because an open system should be designed based on well-established standards that are independent of a single proprietary vendor. Component-integrability is required for the portability and incremental expandability of any open system. For example, if temperature-compensation control is needed, the system should be able to integrate existing results (i.e., temperature sensors, thermal models, compensation algorithms) from thermal compensation research into the controller with minimal effort.

In addition to the openness requirement, an OAC must provide guaranteed real-time performance

The work reported in this paper was supported in part by the National Science Foundation under Grant DDM-9313222. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the NSF.

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 machine status, several levels of control algorithms, and controlling actuators. Some of these tasks are periodically executed and others aperiodically. However, all of them must meet certain timing constraints. In a real-time system, such as a manufacturing system, it is sometimes meaningless to monitor/control the process 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 response time for tasks at different levels of hierarchy and real-time interfaces between 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 these requirements. Section 2 describes the hardware configuration of UMOAC, and Section 3 discusses its software configuration. Section 4 introduces our laboratory evaluation system, or the UMOAC testbed for short.

2. Hardware Configuration

The UMOAC (University of Michigan Open Architecture Controller) is designed to meet two basic requirements: *openness* to meet the need of integrating advanced machine monitoring and control technologies, and *real-time operation* to guarantee response times of tasks at different levels of abstraction between task components in an advanced manufacturing system.

The base hardware configuration of UMOAC is a distributed system in which processing nodes are connected through a real-time link/bus. This distributed system enables the use of 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 of their size and functionality, they operate within a unified software hierarchy and maintain communication compatibility with real-time guarantee. To build a heterogeneous configuration while keeping vendor-neutrality, no specific hardware platform is defined for

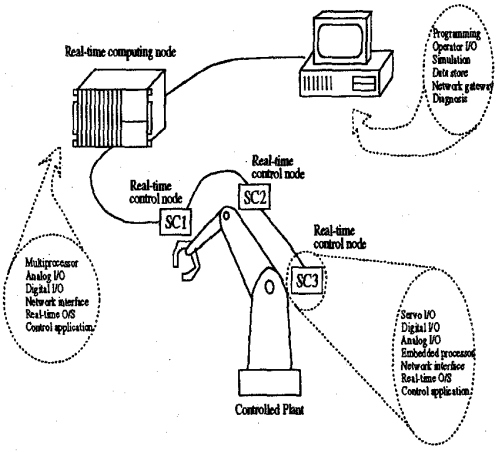

Figure 1: UMOAC hardware configuration

the UMOAC. However, each processing node is based on an industry standard architecture and built with standard off-the-shelf components such as the VME-bus. 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 tasks such as real-time data-logging, diagnosis, scheduling, and controls. The real-time control node performs fine-grain real-time tasks including servo-level control and data-acquisition.

In a distributed system like 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 bounded time, UMOAC adopts the CAN (Controller Area Network)[10] as a real-time communication link between processing nodes. Because it provides a short 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 shows better performance in meeting real-time bounds than existing communication protocols. We proposed a MTS (Mixed Traffic Scheduler) [11] to support periodic, sporadic, and non-real-time messages over a single CAN. Our simulation of real machine-control tasks has shown MTS to outperfom DM (Deadline Monotonic) in handling high-speed real-time data.

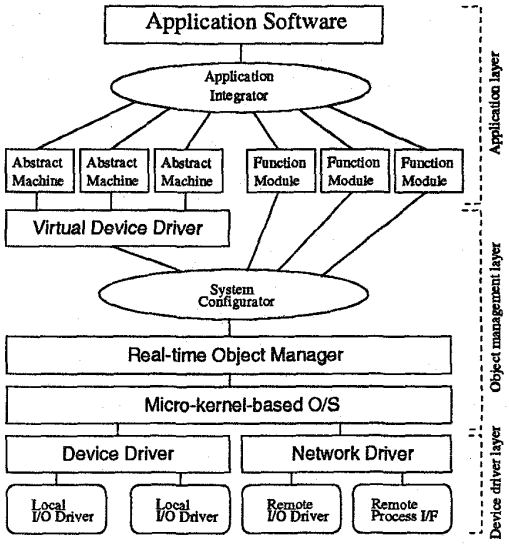

Figure 2: UMOAC Software hierarchy

3. Software Configuration

To enable the writing of highly portable application programs, which should be completely isolated from the hardware configuration, the software hierarchy of UMOAC consists of three major layers: (1) application 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 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 can be reused or extended for other applications.

An abstract machine model 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 pre-defined response time. Since an application program as well as functional modules, interact only with the abstract machine models, they are isolated from the hardware. This isolation enables the modules to possess modularity and reusability.

The second software layer of the UMOAC, the object management layer, consists of virtual device driver, system configurator, real-time object manager, 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 the 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 writing 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 persistency and configuration definition. The real-time object manager 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. Because both local and remote I/O drivers provide the same interface protocol, remote data through CAN can be handled just like local data.

4. Evaluation

To evaluate our design, we have been building a testbed, shown in Figure 3, which consists of a 6-axis CNC milling machine, commercial CNC controller (Robotool), PC-based CNC, VMEbus-based UMOAC, and CAN-based UMOAC. The current testbed has evolved from the activities in the University of Michigan CNC Laboratory, where over the years research on the next generation CNC controllers was conducted [12]. 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 current hardware platform of UMOAC is configured based on VMEbus, which consists of Xycom-486

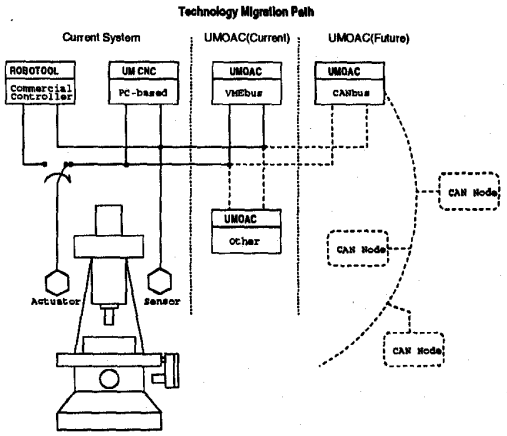

Figure 3: Hardware configuration of the UMOAC testbed

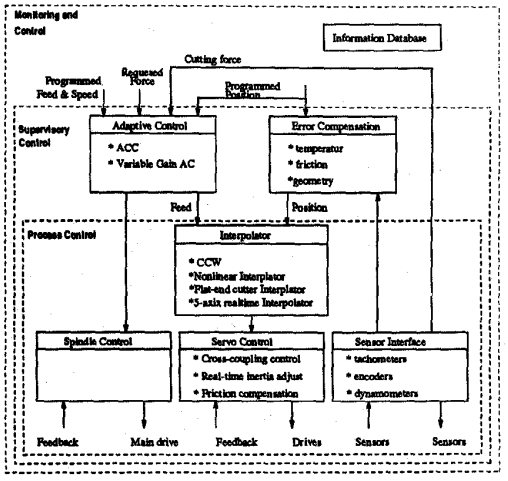

Figure 4: Software configuration of testbed

CPU boards and multiple I/O devices. The object manager is implemented on the top of QNX real-time operating system and all of the application program is written in C++ language.

The UMOAC testbed 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 provided the users with access to data from 16 sensors, including tachometers, digital encoders (rotary and linear), motor current sensors and a spindle power sensor.

Functionally, an OAC for multi-axis machining should provide the multi- level, hierarchical configuration presented in Figure 4. The information database and the monitoring and control routines are divided into specific modules, which provide data integrity. The system's modularity simplifies installation, maintenance and upgradability of the software. Monitoring and control modules have access to the information database, real-time measurements, near real-time measurement and each other. The information database may also be dynamically updated.

The main issue in developing controllers for machining processes is the relative complexity of the system. While research regarding control components involved in such applications, such as servo or process controllers, is well developed, much of the research results are not implementable, mostly due 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.

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. 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.

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: chatter detection and suppression, force control, tool wear rate constraint, surface finish control. 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 feedrate to maximize the productivity given tool wear rate constraints. The tool wear constraint rate module provides the force controller with the current tool wear rate constraint values. The surface finish controller adjusts the feedrate 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 be of contradictory nature. Similarly, the actions of the surface finish module and force controller have to be coordinated during the finishing pass.

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 a 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 the UMOAC evaluation system.

References

[1] G. Pritschow. Automation technology - on the way to an open system architecture. *Robotics & Computer-Integrated Manufacturing*, 7(1/2):103–111, 1990.

[2] G. Pritschow, Ch. Daniel, G. Junghans, and W. Sperling. Open system controllers - a challenge for the future of the machine tool industry. *Annals of the CIRP*, 42(1):449–452, 1993.

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

[4] Manufacturing Technology Directorate Wright Laboratory, Air Force Materiel Command, WPAFB, Ohio 45433-7739. *Next Generation Controller Specification for an Open Systems Architecture Standard*, September 1994. WI-TR-94-8033.

[5] Frederick M. Proctor, Brad Damazo, Charles Yang, and Simon Frechette. Open architecture for machine control. NISTIR-5307. Technical report, National Institute of Standards and Technology, Gaithersburg, MD 20899, December 1993.

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

[7] David B. Stewart, Richard A. Volpe, and Pradeep 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.

[8] Victor B. Lortz and Kang 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.

[9] Y. Altintas and W. K. Munasinghe. A hierarchical open-architecture CNC system for machine tools. *Annals of the CIRP*, 43(1):349–354, 1994.

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

[11] Khawar M. Zuberi and Kang 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.

[12] A. Galip Ulsoy and Y. Koren. Control of machining processes. *Journal of Dynamic Systems, Measurement and Control*, 115(2):301–308, June 1993.