An NSF Engineering Research Center for
Reconfigurable Machining Systems
College of Engineering
The University of Michigan

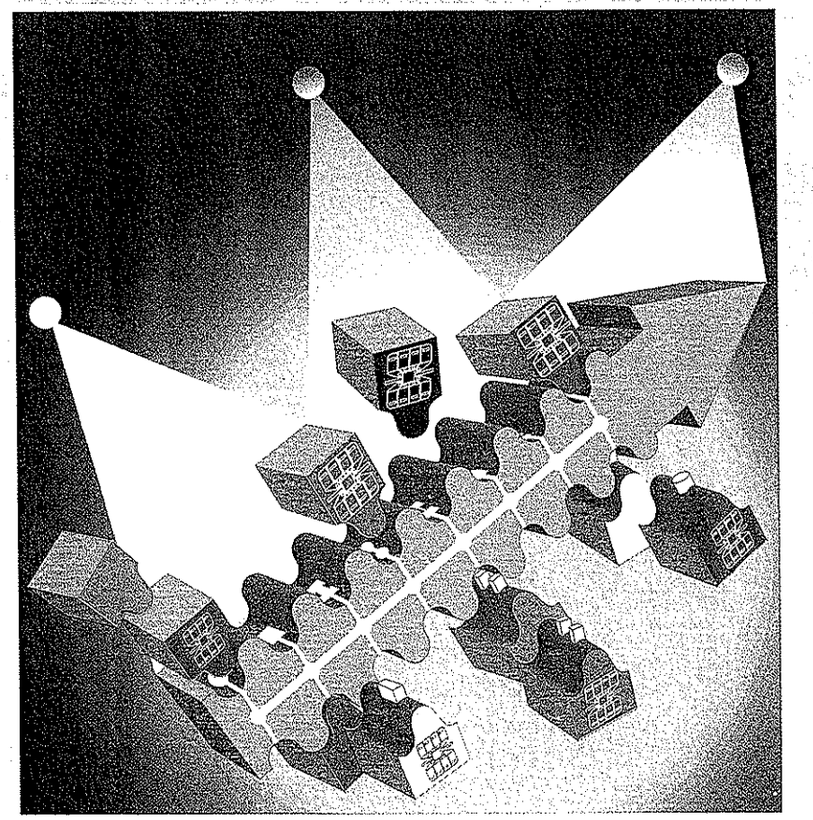
ITIA - CNR
Institute for Industrial Technologies
and Automation

Institut für Steuerungstechnik der Werzeugmaschinen
und Fertigungseinrichtungen
Universitat Stuttgart

Consiglio Nazionale delle Ricerche

Open Architecture Control Systems

Summary of Global Activity



Open-Architecture Controllers for Manufacturing Systems

Yoram Koren
Professor and Director,
NSF Engineering Research Center
for Reconfigurable Machining Systems
The University of Michigan, Ann Arbor

Rapid, controlled-cost response to global market demands will be the cornerstone of manufacturing competitiveness in the 21st Century. Manufacturers are addressing this need by installing more flexible and agile production facilities. However, these present manufacturing systems are expensive and are not designed to be changed or upgraded during their life-time. In addition, manufacturing processes and equipment must be designed to take advantage of new process technologies and to adapt to tighter product specifications. But, again, the current flexible and agile systems do not address this need.

This challenge can be addressed to a significant degree by a new type of manufacturing system, the reconfigurable manufacturing. Such a system can be created using basic process modules –hardware and software– that can be rearranged quickly and reliably. To enable reconfigurable systems, the structure at the system level, the machine level, and the control level must be open to enable integration of new modules and new functionalities. New functionality may vary from installing a new tool magazine on a production machine, to adding a new sensor that monitors part dimensions and sends calibration signals to the machine to enhance precision.

In general, integration of new functionalites may (i) enable production of a new product on an existing manufacturing system, or (ii) improve product quality and increase system productivity. Upgrading or adding functionalities to manufacturing systems may require changes at the system level and changes in its components, including:

- Process and tooling
- Mechanical hardware
- Measurement and control (M&C).

Open-architecture control deals with the creation of a new generation of control systems, that are important for the creation of reconfigurable manufacturing. The Open Architecture Controller (OAC) allows for the integration of new M&C functions into existing control systems, and includes both hardware and software issues.

Definitions

The major advantage of OAC is that it facilitate the implementation of new technology in the M&C domain. New M&C technology may

- enhance part quality (by dimensional measurements and other sensing techniques),
- shorten diagnostic time (by integrating process monitoring and diagnostic functions),
- increase machine productivity (by adaptive controls, chatter elimination, etc.), and
- Lower integration cost of discrete logic

There are several definitions of OACs. The IEEE definition is

> ***An open system provides capabilities that enable properly implemented applications to run on a variety of platforms from multiple vendors, interoperate with other systems applications, and present a consistent style of interaction with the user.***

Nevertheless, we prefer to use our own definition, which although not the standard definition in the computing world, it fits the manufacturing domain:

> ***A controller that is designed and constructed for integration of new M&C devices and software modules by permitting access to a given set of internal controller variables.***

The access to internal variables (e.g., the axial position error within a control loop, or input to the interpolator) is important for allowing easy integration of new M&C modules and algorithms as well as providing access to process data. However, it is difficult to maintain integrity and trust in the behavior of a controller when internal variables are accessed directly or algorithms inside a module are replaced. Therefore, to avoid unintended malfunction, only a given set of internal variables is accessible and **a component or a module must be replaced in its entirety**, which means that a user cannot change an algorithm inside a module nor access its internal variable directly.

The obvious question is to what extent a designer who wants to add new M&C hardware or software modules must be familiar with the design and construction of the original controller? To answer this question we must elaborate on two basic types of open controllers: Vendor-Specific and Vendor-Neutral OACs.

Vendor-Specific OAC is an open system designed by a control vendor for future integration of new M&C devices and algorithms that are also designed

by this vendor. Only the original vendor is familiar with the system design, and therefore only this vendor can reconfigure the controller by adding/changing algorithms or, in some cases, specific algorithms may be added by end-users using the vendor's instructions (e.g., machine temperature compensation algorithm). This approach has several drawbacks:

(i) the potential for expansion of the control system and the level of openness are limited by the vision and the basic control architecture constructed by the original controller designer
(ii) end-users cannot utilize the best, cost-effective algorithm/device in the market and are limited to those offered by the original vendor.
(iii) cost controlled by vendor
(iv) end-users must cooperate with the control vendor when integration of proprietary algorithms are needed (e.g., process models) and therefore must disclose the algorithms to the control vendor.

Examples of companies that offer VS-OAC include Fanuc (Japan), Cincinnati Milacron (Ohio), Cimetrix (Utah), Delta Tau (CA), Hewlett Packard (CA), MDSI (Michigan), Siemens (Germany), and Cranfield Controllers (England). A recent article [Proctor & Albus, 1997] compares the controller architectures of several of these companies. There is no common architecture among their controllers, they utilize different operating systems and their own software applications, and they don't use common Application Programming Interfaces (APIs) to interface among software modules. To conclude, **these controllers are claimed to be open only if new components are also designed by the same vendor.**

Vendor-Neutral OAC is an open system designed for future integration of M&C devices and application software developed and produced by any control vendor (i.e., the controller is open for reconfiguration by third parties). The system architectural design is publicly known, and therefore any vendor can add, change and integrate new M&C devices and algorithms into the original controller. This approach has numerous advantages:

(i) the potential expansion of the control system is not limited by a single vendor
(ii) end-users can use the best algorithms/devices in the market and are not limited to those offered by the original controller vendor. There are two important aspects of openness to such use:
adding a new device/algorithm into existing controller, and
swapping an old algorithm with another one of similar functionality (e.g., servo controller for an axis of motion) but with better performance (see Fig. 1)
(iii) end-users can integrate special algorithms without disclosing the proprietary contents of their algorithms to a control vendor
(iv) reduced operator training and maintenance costs as industrial controllers utilizing similar architecture become more compatible

(v) easier integration of individual machine controllers with the factory control system because of openness in the controller communications modules.

To conclude, **to have an acceptable vendor-neutral OAC, it is critical to have an OAC conforming to a published, vendor-neutral architecture, which provides well-defined methods to exchange data or to invoke services during operation and well-defined methods for control reconfiguration. There should be no need to know the design and construction beyond the knowledge that the controller conforms to the published architecture.**

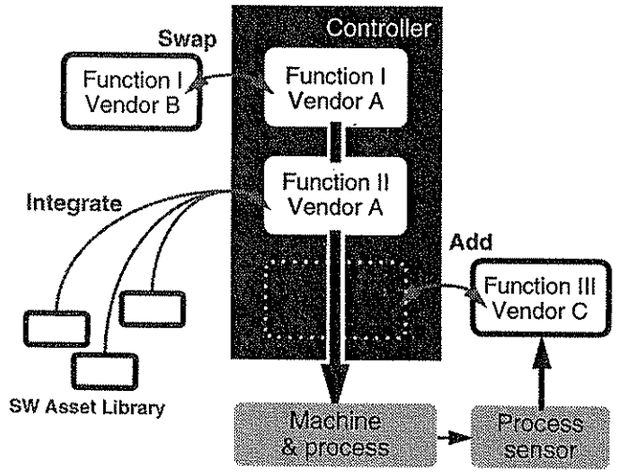


Figure 1. Three aspects of open controllers: SWAP, ADD, INTEGRATE

An example to vendor-specific (VS) versus vendor-neutral (VN) platforms from another domain - personal computers - is the Macintosh versus the Intel-PC. The VS architecture of the Mac limits its expansion. On the other hand, the VN architecture of the PC contributes to its usefulness in many domains. Although a PC may be constructed with devices and modules produced by different vendors (e.g., mother boards, hard disks, modems, etc.), all these modules operate in concert since their interfaces conform to the same architecture, hardware, and software standards in a vendor-neutral environment.

A different aspect of OAC is the reusability of basic modules to build new functions by integrating them into larger modules (see Fig. 1). Examples of such basic modules are an absolute position measuring module that is based on pulses from incremental encoders, or a dynamic model of a machine/robot

axis-of-motion - both may be utilized to develop advanced servo loop control modules where only calibration of their parameters is needed.

Motivation and Development Pace

From the early days of computerized numerically controlled (CNC) machine tools [Koren, 1977] control users have expressed interest in opening the CNC controller for integration of new technologies. A known example of underutilized new technology is adaptive control (AC) for machine tools that, although developed in the 1970's [Tlusty 1974], has not yet been fully implemented. A major reason for this delay has been the complexity of integrating a proprietary AC module developed by one vendor with a proprietary CNC produced by another. Other examples of underutilized mature technologies are cutting tool monitoring for tool wear and breakage, automatic chatter suppression, embedded real-time statistical process control in the CNC, multi-axis real-time 3-D curve interpolators, automatic compensation for fixture errors, model-based control, and graphic user interfaces (GUI) to enhance diagnostics.

However, the penetration of OACs to the market has been very slow. A major reason for the slow development pace of OACs has been resistance from large control vendors, since the development of OAC does not serve their business interests. They prefer to sell new controllers rather than upgrading old ones. They also want to have full control of the market, without letting small companies capture market share by supplying special modules (analogous to modems or sound blasters in the PC market). Therefore, the pioneering R&D efforts in the OAC field were made by the academic community.

Nevertheless, the continuous pressure of end-users, especially from the automotive and aerospace sectors (lead by General Motors and the US Air Force, respectively) eventually changed this trend. OAC has recently became a technology that all end-users and machine builders alike want to adopt as soon as it becomes mature.

U-M OAC - First-Generation

One of the first PC-based OACs was developed for a 5-axis CNC milling machine at the University of Michigan (U-M) by the author and his students during the years 1986 through 1993 [Koren and Lo, 1991 & 1992; Koren and Lin, 1994]. This control system, based on Intel i486/33MHz computer running DOS, utilized the PC microprocessor to perform all control functions as well as the GUI. The controller provided all the basic functionality of conventional CNCs. To connect the controller with the machine, I/O cards were inserted into the PC bus to interface the 5 axial motors, several discrete inputs (e.g., limit switches), and 15 sensors (encoders, tachometers, cutting force/torque

sensors, spindle speed and spindle power monitors). The computer addresses of all these I/O devices and their utilization procedures were documented and known - which created *the basic condition for system openness.* Any student who wanted to utilize the system for the development of new control algorithms could use any subset of the sensors that fit the desired application. Software applications were written in C. This version of the open-controller was working in the Fall of 1987.

The creation of the second stage of openness in this controller was motivated by student needs, as explained below, rather than by trying to develop a controller that fits an OA philosophy. During this period (1986 - 1993) the author supervised several graduate students who developed, implemented, and tested on this machine tool system the following control algorithms:

- Feedforward & Feedback Servo Controller
- Cross-Coupling Servo Control
- Fuzzy-Logic Servo Control
- Model-Based Friction Compensation
- Backlash Sensing & Compensation
- Machine Geometry (X,Y,Z) Compensation
- Three-Axis, Linear & Circular Interpolators
- Five-Axis Real-Time 3-D Curve Interpolator
- Surface Interpolator
- Adaptive Control (based on spindle power and cutting force sensing)
- Tool Wear Estimation (based on cutting force sensing)

The student who developed innovative 3-D curve interpolators did not want to spend time to also develop servo controllers, but would rather use existing ones to demonstrate the interpolation results, and vice versa; the students who developed the Model-Based Friction Compensation (US Patent 5,374,884, 1992) and the fuzzy logic servo controller had to compare performance of their algorithms with other servo algorithms and needed, for demonstration purposes, to quickly swap algorithms with similar functionality [Lo, 1992]; the student who developed the adaptive control had to use available interpolator and servo-loop controls to test the results. The environment was such that each student had not only to be familiar with the work of the others, but to use it and interface his own algorithms with those created by other students. This neutral openness of the controller and its modular structure created the first-generation U-M OAC [Koren, Park, and Lo 1993].

Classification of M&C Functions

The interfacing requirements described above as well as the need to rapidly swap algorithms with similar functionality motivated the design of the U-M modular controller depicted in Figure 2. Levels were determined based on common input and output dimensions (e.g., position is the output of all types of interpolators) or on common functionality (e.g., monitoring). Each block in the "M&C Modules" section of Figure 2 describes an algorithm or an application program (AP) with defined software interfaces. The computer contains a library of APs (several APs depicted in Fig. 2 were developed at U-M more recently), and each developer can select a desired combination of modules that fits the application. *The library of modules may be classified by function.* Figure 2 shows 11 basic classes of M&C functions:

1. **Discrete Events** (on-off commands, logic circuits, interfaces with PLC, etc.)
2. **Servo Control Loops** (control axial motions and spindle speed)
3. **Interpolation and Trajectory Generation** (coordinate motion of a group of axes)
4. **Compensation for Machine and Tool Errors** (e.g., machine temperature)
5. **Adaptive Control** (e.g., control feeds & speeds to increase productivity)
6. **Process Models & Real-Time Simulators** (compare simulated & real performance)
7. **Monitoring** (of machine and process variables)
8. **Diagnostics of Machine and Process** (of quality & reliability problems)
9. **Human Machine Interface & Graphic User Interface** (present information)
10. **Translators for Part Programs and Task Programs**
11. **Communications with other factory computers** (e.g., cell level)

Each of these functions contains a set of modules, as depicted in Fig. 2. The number of "discrete event" modules might be the dominant when the synchronize control of multiple devices in large systems is needed. Modules in each class have input and output variables of similar type (e.g., position; see Table 1 for details). In some cases modules may be used simultaneously to control a device or a system ("*add*"), while, in other cases, the designer must select one module from a class ("*swap,*" as in the Servo and Interpolation classes; see [Koren and Lo, 1992]). Let us elaborate on the issues of modularity and module interfaces.

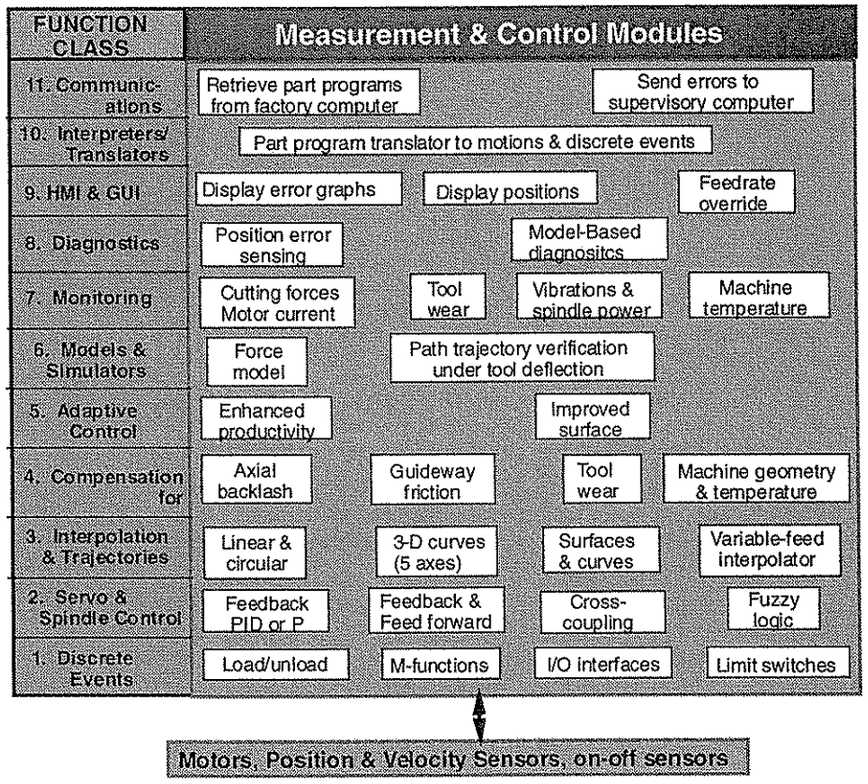


Figure 2. Library of U-M M&C modules classified by functions.

Modularity. A key enabler to open architecture is modularity. The various M&C functions must be designed in a modular fashion. A consequential logical question is

"What is the optimal size (or basic functionality) of a module in OAC?"

In determining the module size, there is an obvious trade-off between the degree of openness and the cost of integration. Smaller modules (i.e., a lower level of granularity) enable a higher level of openness and more options, but increase the complexity and integration costs (and may even deteriorate the real-time performance). The control community does not have a definite answer to this issue. There are researchers who believe that this level of classification is not always adequate. Based on our experience, however, we believe that Fig. 2 shows a practical functional classification of control modules that is generic enough to be adopted by the control community and be utilized by other OAC developers to define APIs.

API. Defining a comprehensive set of standard Application Programming Interfaces (APIs) is a key enabler for vendor-neutral OAC. An API is an "envelope" around a control function that defines its interface to other modules. Each module consists of a function and an appropriate known API (see Fig. 3). The API specifies the function name, calling sequence, input variables, and return variables. The API gives third parties all the information needed to integrate their algorithms into an OAC. (Note that if the OAC granularity is too small, time is wasted on calling many APIs and the resulted real-time performance might be deteriorated.) A project called "TEAM", which is described later in this paper, is currently concentrating on defining a comprehensive set of acceptable APIs.

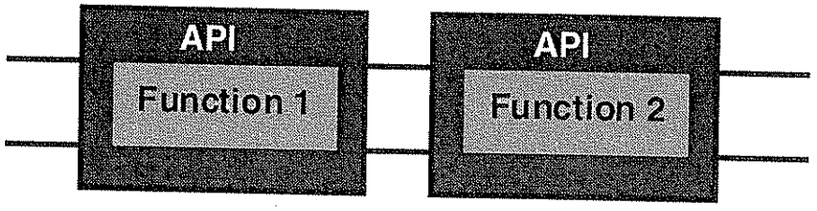


Figure 3. Two modular CNC functions are interfaced through their APIs

U-M OAC - Recent Developments

While providing controller openness necessary for research, the first-generation UM-OAC exhibited a number of drawbacks. Its performance would vary, depending on the programmer's skills; for example, execution times of the subroutines are a function of the length of the code. Therefore execution of critical real-time tasks could not be strictly enforced. This issue became even more important with the computational load increased by a growing number of involved control routines and their complexity. Moreover, the controller was based on a single processor, relying on interrupts, and practically excluded applications where multitasking would be needed

For these reasons the original U-M OAC has been under continuous development (1993 - 1997), this time with the intention to develop and evaluate an innovative OAC technology [Pasek et al, 1995]. The main researchers who participated in this effort (in addition to the author) have been G. Ulsoy, K. Shin, S. Birla, Z. Pasek, and S. Jee [Jee & Koren, 1994; Koren & Lo, 1993, Pasek et al., 1995; Ulsoy & Koren, 1993].

Three generations of U-M OACs are shown in Fig. 4. The present version is based on a multi-tasking, object-oriented organization, heirarchical decomposition of functionality [S. Birla and K. Shin, 1995] into modules.

The computations in the present UM control system are distributed across a number of modules are structured into cooperating independently executing tasks, allocated across that communicate with each other. The system is functional and run on three computers, as follows:

- one computer handles supports the user interfaces with soft real-time performance, and non real-time interactions with remote computer systems.
- A second computer is used for one contains overall task coordination, multi-axis motion coordination, and individual servo-controlled motion of each axis the servo controllers for all five axes of motion.
- A third computer is used for one performs cutting force data acquisition, sampling at 0.1 ms sub-millisecond intervals, for and handles the adaptive control.

The controller architecture is defined in has been developed through a domain engineering process of evolving requirements specifications, evolving from specific case studies to a domain definition by applying existing domain knowledge. [Birla, 1997]. System design rules and constraints have been defined for applications with servo-controlled motion. This research has also identified promising avenues for long-term, mid-term, and short-term research. Present research effort in this direction is proceeding under the supervision of K. G. Shin and C. V. Ravishnkar.

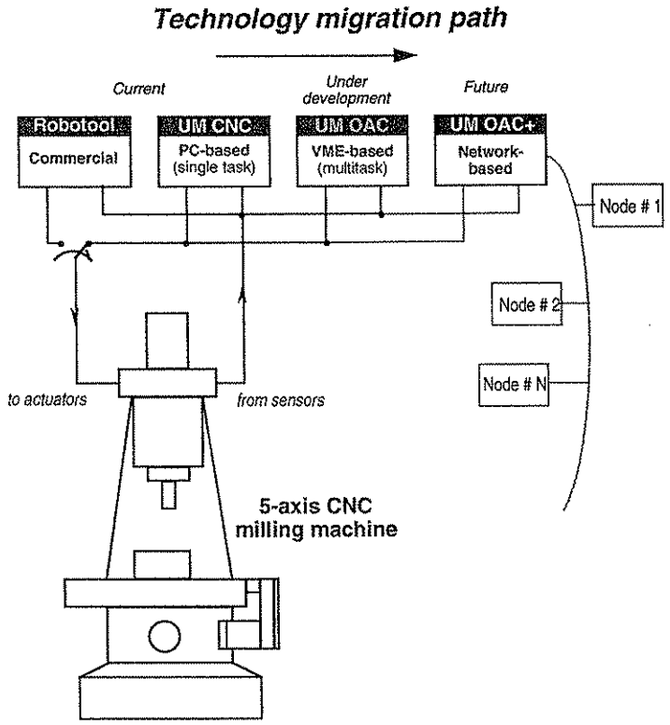


Figure 4. Three generations of U-M OAC

Controller Architecture. A controller architecture based on the functionality shown in Fig. 2 is depicted in Fig. 5. This drawing shows possible interactions among the various M&C functions in controlling a machine tool. Understanding of these interactions is an important requirement for distributing computation tasks on several computers when designing the controller. At the system level, the management of discrete events may become the dominant control requirement.

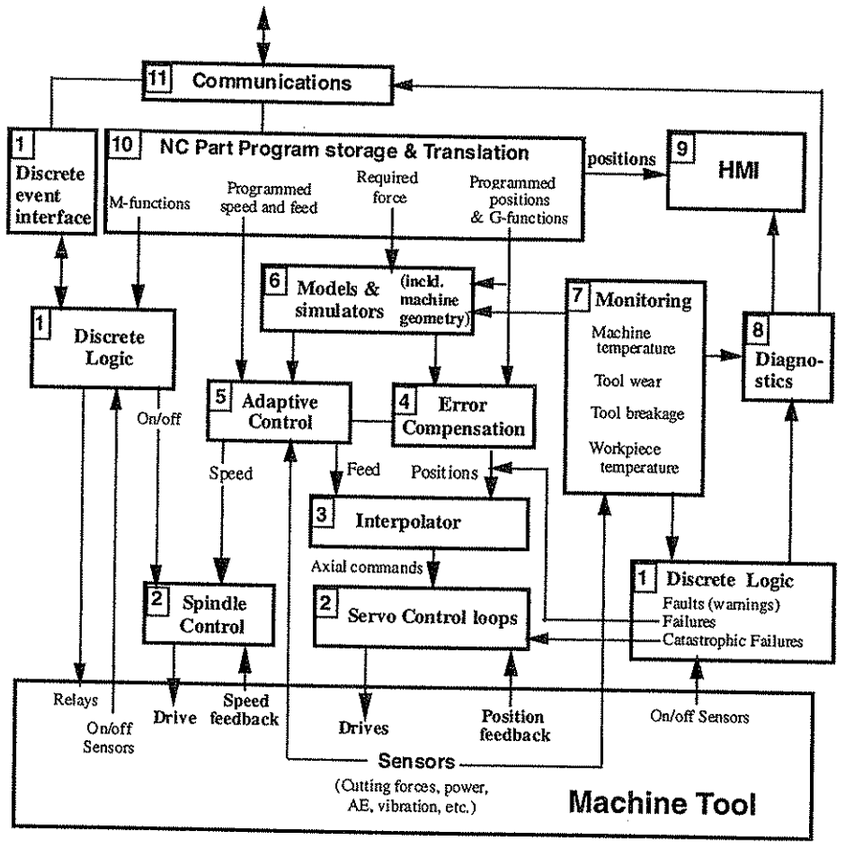


Fig. 5 Hierarchical Levels in CNC Controllers (by Yoram Koren)
(Using this diagram requires a written permission from the ERC/RMS)

Other Efforts to Develop OACs

Other major research efforts in the area of open architecture systems include the following:

• The **MOSAIC** (Machine Open System Advanced Intelligent Controller) project in the late 1980's [Greenfield et al, 1989; Wright 1990] was one of the first attempts at establishing open architecture for machine control. The development of this system, was motivated by similar needs that initiated the development of the U-M OAC — the desire to utilize in a single controller several independent algorithms developed by this team in cooperation with other researchers. These algorithms were developed originally on different hardware platforms, using different operating systems and different computer languages. MOSAIC is based on a Workstation/VME-bus/Real-Time Unix/C architecture: it provides all of the expected functionality of a conventional "closed" CNC. That is CNC part-programmers can interact with the MOSAIC system from any APT-based language or any CAD/CAM software with the result that the files generated through MOSAIC are processed into machine-level commands that replace the conventional G-code format and provide exactly the same functionality as G-codes.

• The **Next Generation Controller** (NGC) project (initiated by the US Air Force) was the most influential effort to take the OAC development "off the ground." In the summer of 1987, in a workshop organized in Dayton, Ohio, a draft summarizing the basic requirements for OAC was written by R. Kegg, S. Birla, and the author, and later used by the Air Force for a bid. Martin Marietta (MM), that won the bid, started in 1988 to work on the contract which became known as the Next Generation Controller (NGC) project. With a budget of $20 million, during four years (1988 - 1992), MM tried to develop a widely acceptable VN OAC [Anderson et al., 1993]. The only end result was, however, a document summarizing the Specifications for an Open System Architecture Standards (SOSAS) [martin Marietta, 1994].

• The Air Force continued the project on a much smaller scale ($1 million) by developing an **Enhanced Machine Controller** (EMC) with a joint cooperation of MM and the National Institute for Standards & Technology (NIST). In the EMC project, an open machine tool has been implemented based on the NGC/SOSAS. The contribution of NIST was the development of a Reference Architecture [Proctor & Michaloski, 1993; Proctor et al., 1993], which is being utilized by the "TEAM" project described below.

• A push towards the development of OAC was a document published by the US automotive industry which summarizes the **Requirements of Open, Modular Architecture Controller** [OMAC, 1994]. The writing of this document was lead by J. Yen (GM), C. Bailo (GM), R. Furness (Ford), and W. Haukkala (Chrysler).

• OMAC established the basis for a Department of Energy (DOE) led recent project called Intelligent Closed Loop Processing (ICLP), as part of a larger program, called **"TEAM"** (Technologies Enabling Agile Manufacturing. TEAM.), which is currently the most promising effort in the US to develop VN OAC. Three DOE labs (LLNL, Los Alamos, and Oakridge) collaborated with GM, Ford, Chrysler.
TEAM is conducted jointly by the Big-3, NIST, and U-M in developing, and three DOE Laboratories. TEAM is developing a comprehensive set of APIs, modularizing in the object-oriented paradigm. for different Certain control modules are which is being validated on three different testbeds located at NIST, U-M, and ICON LLNL. The latter is working with ICON, (a private company,) for the commercialization of a controller utilizing a subset of these APIs [TEAM, 1996].

• Other research projects like the **Chimera** project at Carnegie Mellon University [Stewart et al., 1993], the **Hierarchical Open Architecture Multi-Processor Motion Control System** (HOAM-CNC) at the University of British Columbia [Altintas & Munasinghe, 1994], and the **multiple-pass strategies** on OACs [Taltz et al., 1995] have demonstrated a variety of approaches to the OAC. For example, the HOAM system at UBC utilizes PC with Windows NT for user interface and a DSP (inserted into the PC bus) for motion control..

• In parallel to the efforts in the US, an European consortium called **OSACA** (Open System Architecture for Controls within Automation systems) consisting of 3 universities and 10 European companies received in 1993 a 3-year grant of approximately $15 million to develop OAC [Pritschow et al., 1993]. Under the leadership of G. Pritschow of the University of Stuttgart, the consortium developed standardization for networking and application software for OAC.

• In Japan an OAC consortium was formed in January 1995 with cooperation between academia and industry (25 companies, including Mitsubishi Electric, Toshiba, Toyoda Machine Works, Yamazaki Mazak, IBM Japan, and Sony, and 12 academic institutes). The consortium, called **OSEC** (Open System Environment for Controllers) is in the process of developing APIs and standard interfaces for open controllers.
The major efforts in the late 1990's are the TEAM/OMAC in the USA, OSACA in Europe, and OSEC in Japan.

Legal and Technical Issues

Despite the numerous advantages, vendor-neutral OAC carries some unresolved legal and technical issues that may impede its implementation. These include

- Legal liability

- Ensuring technical performance after modifications because of difficulties in
  - honoring timing constraints
  - correct integration of new modules
- Loss of production during installation of new functions.

The legal issues regarding a loss or a damage that occur because of modifications in the original controller by a third party, may become a major impediment to implementing OACs. These reliability and safety issues include questions such as:
Why did the damage occur? Is it directly or indirectly related to the modification?
Who is responsible for a loss in productivity or a damage? Is it the third party that performed the modification, the end-user who ordered it, or the original controller vendor?

There are no definite answers to these questions, but some end-users have concerns that eventually they will be found liable (at least in the US). Therefore, these end-users may prefer the less advantageous vendor-specified OAC that will restrict the liability issues.

In addition to legal issues, also the technical issues in implementing VN OAC should not be overlooked. Since the CNC controller operates in real time at fixed sampling periods, honoring timing constraints must be considered at the controller reconfiguration stage [Koren et al., 1996/1 and 1996/2]. The real-time requirements of CNC are relatively stringent compare to those of other domains (e.g., aircraft control). The integration of additional algorithms that are executed during the motion of the machine axes, requires an increase in the sampling period, T. This brings up two consecutive issues: (i) To maintain part precision the increase in T may require a corresponding decrease in the velocity (i.e., feedrate), and (ii) software integration tools that guarantee the timing constraints and the fitting of modules, do not exist. Therefore, there is a need to develop a suite of software tools to aid in the integration of M&C software modules into control systems customized to best match a particular application. The Engineering Research Center for Reconfigurable Machining Systems (RMS) at U-M is conducting research in this direction.

Conclusions

Solving the technical issues requires a new direction in R&D of industrial controls. We envision that industrial control will evolve in a manner similar to that of the desktop computer hardware and software. There will be companies producing "hardware platforms" and companies developing generic software tools for integrating control modules (equivalent to word processors) to enable users to integrate their proprietary modules into configurable user-tailored controllers. Out of this will grow a major industry of

system integrators and software houses established to integrate users' modules, and industries to produce new modular machines and controllers, giving a major competitive advantage to all U. S. manufacturing industries.

Acknowledgment

This research has been sponsored by the National Science Foundation and the Engineering Research Center for Reconfigurable Machining Systems.
The author would like to thank Dr. Sushil Birla for his useful comments.

References

Altintas Y., Munasinghe W. K., 1994, "A Hierarchical Open-Architecture CNC system for machine tools," *CIRP Annals*, Vol. 43, no. 1, pp. 349-354, 1994.

Anderson B., Cole J., Holland R., 1993, "An Open Standard for Industrial Controllers," Manufacturing Review, Vol. 6, No. 3, pp. 180-191

Birla S. and K. Shin, 1995, "Software engineering of control systems for agile manufacturing: an approach to lifecycle economics," *Proc. of the 1995 IEEE International Conference on Robotics and Automation.*

Birla S., 1997, "Software modeling for reconfigurable machine tool controllers," *PhD thesis,* the University of Michigan.

Greenfeld I., Hansen F. B., Wright P. K., 1989, "Self-Sustaining, Open-System Machine Tools," *Trans. of the 17th NAMRC*, Vol. 17, pp. 304-310

Greenfeld I. and Wright P.K., "A Generic Specification for an Open-System Machine Controller", ASME, Winter Annual Meeting, Volume on Advances in Manufacturing System Engineering, PED-Vol. 37, pp. 63-76, San Francisco, CA, Dec., 1989.

Jee S. and Koren Y., 1994, "Friction Compensation in Feed Drive Systems Using an Adaptive Fuzzy Logic Control," DSC-Vol. 55-2, *Dynamic Systems and Control, ASME*, pp. 885-893

Koren Y., 1977, "Computer-Based Machine Tool Control." *IEEE Spectrum,* March, pp.80-83.

Koren Y. and Lo, C., 1991, "Variable-Gain Cross-Coupling Controller for Contouring." *CIRP Annals*, Vol. 40, No. 1, pp. 371-374.

Koren, Y. and Lo, C.,1992, "Advanced Controllers for Feed Drives." (Presented as a keynote paper.) *CIRP Annals*, Vol. 2, pp. 689-698, October.