COMPARISON OF THREE DIFFERENT OPEN ARCHITECTURE CONTROLLERS

János Nacsa

Computer and Automation Research Institute (www.sztaki.hu)
Kende u. 13, H-1111 Budapest, Hungary, nacsa@sztaki.hu

Abstract: Since the late 80s there are a growing need of the machine tool builders and the end users to have open architecture controllers. Different levels of openness can be defined analyzing the available research results and products. The main initiatives leading the current research in these efforts are the European OSACA, the American OMAC and the OSEC from Japan. A complex comparison of these projects and their goals and achievements is presented in this paper. It is stated that in their present form it is not possible to merge easily the three results to get a unified and potentially worldwide accepted open architecture controller.

Keywords: open architecture controller, CNC, reference architecture, application programming interface.

1. INTRODUCTION

Manufacturing has constantly been a technological domain, in which the industry was driven to apply the current high-tech from the computer and the control area as well. It is not surprising, that the Open Systems concept has also diffused into the manufacturing area, and factory managers are often referring the open manufacturing systems. The terms and definitions are far less exact than the terms applied in the operating systems environment, but by now, the change of the global manufacturing paradigms are directing our focus on the key user aspects of openness.

The booming market for industrial automation led to a vendor dominating situation: dozens, or rather hundreds of controller manufacturers (vendors) are developing, implementing and installing different solutions for automation tasks. But several problems are arising because of this situation:

- there are a large number of incompatible products,
- the controllers can not cope with the frequently needed updates,
- the service, maintenance and repair costs are scoring high,
- professional personnel to work with controllers are decreasing in number.

The need for a new and vendor neutral open CNC architecture was emerging at many places around the world. One of the most important work was done from 1992 within the frame of the European project named OSACA (www.osaca.org). Similar efforts are going on in Japan named OSEC under the IROFA Consortium (www.sml.co.jp/osec/), and in the U.S. within the OMAC projects (www.arcweb.com/omac) where many earlier American research results were collected. The first part of the paper introduces these efforts.

In the second part a general methodology is described how the open controllers are comparable. Based on this an assessment and a comparison is done, how these three initiatives are similar/different. The main question is the following: if is it possible to merge the different specifications into a unified architecture?

The analysis shows that the answer is: impossible. A suggestion is done finally on what basis it is worth to continue the work for a worldwide accepted open architecture controller.

2. OVERVIEW OF VENDOR NEUTRAL OPEN ARCHITECTURE CONTROLLERS

Fig. 1 shows the three different ways of openness in a controller architecture.

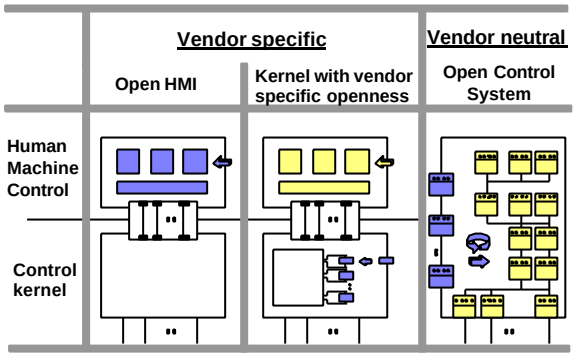

Figure 1: Different types of architectures of open controllers [OSACA, 1997]

The first architecture means an open human machine interface (HMI), that is a must nowadays. Most of the vendors (e.g. GE Fanuc and Siemens) offer these features using a PC/Windows based environment coupled to the NC kernel which solves the motion control tasks in real-time.

In the second type of open architecture the vendor offers interface also to the NC kernel or -at least - to insert user specific filters (e.g. Sinumerik 840C can be set-up in this way with special agreement from Siemens). Many smaller vendors have PC based 'open' solutions where an add-on DSP card makes the motion control. Others run a real time operating system parallel to the MS Windows on the same single processor. In both cases users have an API to build their applications based on the vendor specific open platform.

The third type of open architecture is vendor neutral. In this case a sort of joint consortium defines all the interfaces of the different controller modules including the real time parts (e.g. motion control, axis control, PLC functions). The three most important vendor neutral initiatives are shortly introduced in the following.

2.1 OSACA (Open System Architecture for Controls within Automation Systems)

In the OSACA project [Pritchow, 1992] the different partners (universities, control vendors and machine tool builders) wanted to develop a vendor neutral open architecture controller. The final result has two basic elements:

- A general application programming interface (the upper layer on the Fig. 2), that is independent form the information infrastructure (hardware, operating system and communication channel), where the so called OSACA platform is available. Currently about 6-8 different OSACA platforms are available. This API has further sublayers and manages different types of items (events, variables, actions) to communicate with.

- A reference architecture that defines the basic OSACA modules (the little boxes on the Fig. 2) with their function specific items. In OSACA the NC kernel functions were developed very precisely (motion control, axis control, spindle control, motion management control), while the others (e.g. PLC like logical functions) are still undefined.

An easy to use configuration system supports the collection of the necessary application objects for a given controller and it also manages the start-up phase of the controller.

Many good and successfully demos, exhibitions and pilot applications at industries, as BMW and Mercedes proved the concept and some advanced application objects were also developed (e.g. [Nacsa, 1997]).

The main problem with the OSACA is that it has not got any advancement since 1998, where the EC project stopped officially. As it was designed in the early 90s, some software solutions are rather dated by now (e.g. naming conventions, usage of variables).

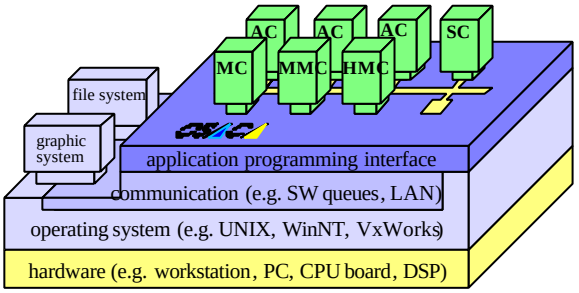

Figure 2: OSACA platform with the basic application objects [OSACA, 1997]

2.2 OSEC (Open System Environment for Controllers)

In Japan some important vendors established a workgroup to develop a Japanese open controller. Later many similar efforts dealing with the factory automation were merged together under the umbrella of JOP (Japan FA Open Systems Promotion Group).

OSEC is focused only on PC platform and Windows environment and it does not allow distributed control.

In the OSEC they defined a 7 layer reference model (similar to the OSI network layers). They have paid attention to the programming interface between the different layers. They call it Message Coordination Field (see Fig. 3) and defined different C functions to each layers. It is sensible that only a limited number of the functions are mandatory in an OSEC controller.

Later in JOP they published another API between the NC kernel and the human machine interface (HMI) that is called PAPI. Unfortunately the OSEC and PAPI is not really harmonized (e.g. not easy to map the functions of the two APIs even they have the same meaning logically).

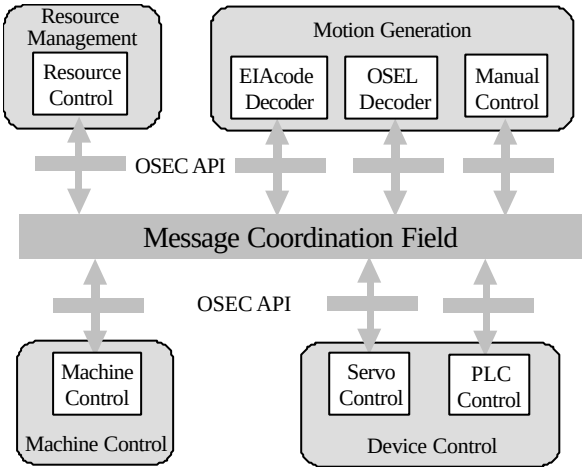

Figure 3: OSEC Reference Architecture [OSE, 1998]

2.3 OMAC (Open Modular Architecture Controller)

In 1994 the three big American car manufacturers published a white paper [Chrysler, 1994] about their requirements of the future open and modular controllers.
OMAC has not defined a fix reference architecture, but a set of modules to build up with them different types of controllers . In the current version [OMAC, 1999] of the API 14 complex modules exist. They are specified in IDL (Interface Definition Language), and many of them have subparts. The detailed specification of the modules is getting critical, e.g. the Axis module has more than 10 subparts with more then 400 methods

Fig. 4 shows a controller of a drilling machine where the tight synchronization of the spindle and the axis Z is possible.

There is no direct specification of the information infrastructure of the OMAC based controller, too. At least four prototype realizations are known and all of them have different infrastructures and development environments. Nevertheless industrial pilots are not known.

On the other hand - and it is unique but very important - OMAC supports the internal structure of the modules when defines final state machines (FSMs) within the modules.

OMAC also has started to deal with other related issues (e.g. HMI, real time environment under Windows, new type of NC programming language), but presently no strict schedule exists, so the speed of the development is rather slow.

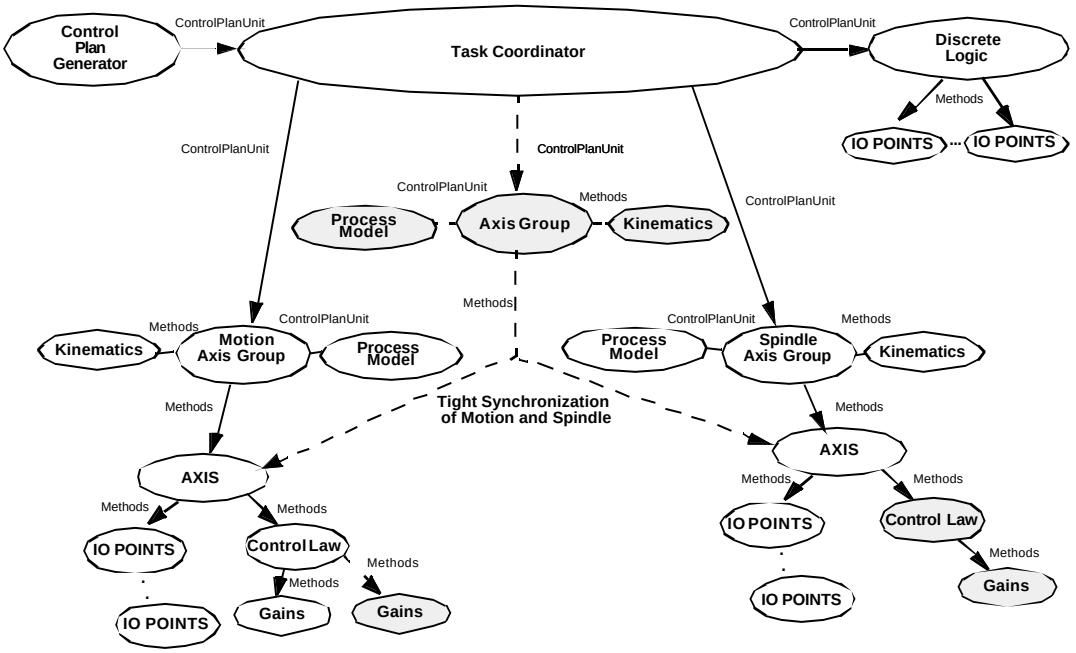

Figure 4: OMAC based modular controller of a drilling machine [OMAC, 1999]

3. COMPARISON OF THE THREE OPEN CONTROLLERS

The final aim of each initiatives is to develop a widely acceptable and vendor neutral open architecture controller. So it is an important question if these efforts are compatible or -at least how they are similar. OMAC has made efforts to use the results of the other two projects, especially in Global HMI API Common Data Model defined in XML schema.
An early comparison of OSACA and OMAC states that there are many common features (e.g. object oriented methodology) in both approaches [Lutz, 1998], but it did not examine the details of the specifications.

According my methodology in a detailed analysis the following should be examined (1) the application programming interfaces in software point of view, (2) the reference architectures and (3) the information infrastructures to see how much common exist in the three initiatives.

3.1 Comparison of the Application Programming Interfaces (APIs)

In Table 1. the main points of the API comparison are shown. Unfortunately the basic ideas (logic) and the levels of abstraction are different. OMAC has been developed in much more details than the others.

	OSACA	**OSEC**	**OMAC**
Logic	Obj. orient.	Functions	Component
Definition	C++	C	IDL
Granularity	OK	Big pieces	Small pieces
Size	~100 items (variables, events etc.)	~ 150 functions	Thousands of functions, many types
Advantage	Size of granularity	Easy to understand	Complete
Dis-advantage	Variables instead of-functions	Not object oriented, too simple	Too complicated

Table 1: Different properties of the APIs

3.2 Comparison of the Reference Architectures

Because of the importance, the most critical parts of the comparison are the reference architectures (Table 2). While all the initiatives have the same purposes, the necessary functionalities should be roughly the same. The different module structures group these functionalities in a certain way. But the main problem of the comparison is the following: these groups are logically incompatible. There is no way to map the module structure of one system into the others. E.g. in one system a given method/function is part of the axis module, while in the others the same one belongs to the motion module.

	OSACA	**OSEC**	**OMAC**
Reference architecture	Exists	Exists	Modules as building box
Named Modules	9	7	14
Specified modules	4 from 9	all	all
Range of potential applications	Acceptable	Acceptable	Wide
Module inner description	Undefined	Undefined	Specified with Final State Machines

Table 2: Features of the reference architectures

OSACA and OSEC does not seem enough detailed to be sure that two realizations of the same abstract module would be really interchangeable.

3.3 Comparison of the Infrastructures

It is not difficult to compare the IT features and needs of the initiatives (Table 3.). Only OMAC is really open towards many software platforms. OSACA needs its own platform to run, so always hard software development is needed to move to a new environment. OSEC is limited in PC/Windows world.

	OSACA	**OSEC**	**OMAC**
Comm. platform	OSACA specific	Specific dll-s	Anything
PC - Intel	Also others	Only	Anything
Windows	Also others	Only	Anything
Prog. lang.	C/C++	C/C++	C/C++/Java
Distributed	Yes	No	Yes
DCOM, Corba etc.	No	No	Yes
Modernity	Outworn	Outworn	Hot

Table 3: Features of the hardware, software environments

3.4 Summary of the comparison

The result of the comparison is not nice. In many ways incompatibility problems were found. It became clear that there is no way to merge simply the results of these activities into a unified open controller.

4. UNIFIED OPEN CONTROLLER

While the needs of an open controller are still here, the presented comparison proved that if we want to use previous results one must choose one of the initiatives, because merging them is very difficult even it is hopeless. It is also clear that it is not possible to build up a controller that is e.g. partly OSACA and partly OMAC based.

The detailed results show that the OMAC initiative is the most promising. But the analysis told that the API of OMAC is too complicated.

A simple suggestion to simplify it is the following: Separate the methods of the OMAC modules into mandatory and optional ones. Keep only the most important ones as mandatory. The following objectives were taken into account during this work:

- The methods belonging to the basic functionality of the module must be mandatory.
- Modules should keep their FMS inner pursuit, so the methods belong to the state transitions are mandatory.
- Reference methods to another modules should be kept also.

A detailed examination of the OMAC axis module [Nacsa, 2001] results that about 80% of the functions may be optional.

5. CONCLUSIONS

The most important open control initiatives were compared and proved that they are incompatible in many ways.
As a solution for the future developments, a simplification of the OMAC API was suggested.

REFERENCES

Chrysler, Ford Motor and General Motors. Requirements of Open, Modular Architecture Controllers for Applications in the Automotive Industry, 1994, White Paper - Ver. 1.1.

Lutz, P: Comparison between the OSACA and OMAC API approaches on an Open Controller Architecture, in: Open Architecture Control Systems, ITIA Series, Vol. 2, 1998, pp.203-208

Nacsa J, G. Haidegger: Built-in Intelligent Control Applications of Open CNCs, Proc. of the Second World Congress on Intelligent Manufacturing Processes and Systems, Budapest, Hungary, 1997 June 10-13., Springer, pp. 388-392

Nacsa J: Intelligent, Open Controllers and their Knowledge Sharing Problems in Manufacturing Systems, PhD dissertation

OMAC API Work Group: OMAC API Set, Ver. 0.23, 1999

OSACA Association: OSACA Handbook, 1997, Stuttgart, FISW Gmbh.

OSE Consortium: OSEC-II Project Technical Report, 1998

Pritchow G, Ch. Daniel, G. Junghans, W. Sperling: Open System Controllers - A Challenge for the Future of the Machine Tool Industry, Annals of the CIRP, 42/1, 1993, pp. 449-452