ENHANCING NUMERICAL CONTROLLERS, USING MMS CONCEPTS AND A CORBA BASED SOFTWARE BUS.

Raymond Boissier[1], Eric Gressier-Soudan[2], André Laurent[2], Lionel Seinturier[3]

[1]GRPI / IUT de Saint Denis. 93206 Saint Denis Cedex 1. France
Tel. +33-1-49 40 61 71 Fax.: +33-1-49 40 61 71
E-mail: boissier@iut-stdenis.univ-paris13.fr

[2]CEDRIC / CNAM. 292, rue Saint Martin. 75141 Paris Cedex 3. France
Tel: +33-1-40-27-25-24 Fax: +33-1-40-27-27-09
E-mail: Eric.Gressier@cnam.fr

[3]LIP6 / Université Paris 6. 4, place Jussieu, 75252 Paris Cedex 5. France
Tel: +33-1-44 27 87 64, Fax: +33-1-44 27 74 95
E-mail: Lionel.Seinturier@lip6.fr

Abstract: *Open Manufacturing Equipment is mandatory to answer the demand of modern manufacturing for distributed monitoring or automation, and to allow fast system reconfiguration. The project is inspired by the reference models defined by the ISO/OSI Manufacturing Message Specification for remote machine tool supervision (MMS) and the Open System Architecture within Control Applications (OSACA). Instead of proposing MMS or OSACA protocols for communication, we propose a unified object-oriented solution based on the OMG-CORBA software bus. Adopting a common object-oriented approach for both remote communication and internal architecture of a NC platform is conceivable. It leads to a design very close to the one proposed in the mentioned standards. The consequences of including a Manufacturing Object Messaging System on the NC design model are discussed. A Java and CORBA based demonstration prototype has been developed. The end of the paper describes how we address real-time constraints on communications.*

Keywords: *Open Manufacturing Equipment, Numerical Control, Object Oriented Design, MMS, CORBA, Industrial Messaging, Real Time Communication, OSACA.*

1 Introduction

Making the various computerised and programmable manufacturing equipment co-operate is an old wish of the manufacturing systems integrators (AMICE-CIMOSA 1993). It needs communication means and standardised protocols and interfaces, something not so easy considering the variety and complexity of industrial equipment such as robots and machine-tools. This task was initiated by General Motors in the early eighties and lead to the ISO MAP/MMS standard (ISO 1990). The MAP/MMS solution was a heavy one and, if conceivable for automotive large factories, was hardly adaptable to small production units for reasons of price, complexity, and rigidity of legacy control systems.

More recently, together with the dissemination of object-oriented modelling and programming techniques, the interest has moved toward modular software architectures for control software, allowing fully distributed control systems. One of the most typical initiatives in that sense is the European OSACA project which specifies how to build an NC application out of so-called Architecture Objects and an unified communication mechanism which binds these objects (Pritschow et Al. 1994) (OSACA 1997).

We consider the physical NC station as a subclass of a generic open workshop station:

- from which an operator can access any relevant production data he may need ;
- that includes a NC as Server Object (corresponding to an MMS provider), which can be accessed from a local or distant client ;
- that numerically controls a local NC device.
- that will be inspired by MMS external functionality and OSACA internal modularity.

The implementation of such an open workstation can be based on an inter-object communication service, more commonly known as *Object Request Broker* (ORB). Provided component object interfaces are well standardised,

the NC system element can become a plug-in component i.e. can be replaced or upgraded without recompilation or regard to the source programming language.

The trail we have been following results from our own former investigations about NC modelling (Raddadi et Al. 1995) and distributed real-time application (Gressier et Al. 1996), from our consideration of the MMS and OSACA proposed models, and from our choice of the *Common Object Broker Architecture* (CORBA), a standardised object-oriented software communication bus.

Our work relies on the use of the CORBA standard proposed by the Object Management Group consortium (OMG 1999a). The general design of our platform would not have been changed if we had built it on different object oriented communication services such as Java Remote Method Invocations (Sun 1998) or Microsoft's DCOM (Grimes 1997). However the advantage of CORBA is being a multi-language non-proprietary open platform. Furthermore, the last version of Java, called Java 2, includes the basic features of CORBA. At the time we started our research this was not the case. Today, it is not a meaningful aspect because our work targets a component based approach, independent from any particular ORB implementation.

Our Open NC approach was originally conceived for those small and medium enterprises (SMEs) devoted to CAD/CAM and machining, with little automation, where highly skilled staff must access on line help and tools to achieve an excellent reactivity and high quality standards. It is expected that Internet development will stimulate the interconnection of SMEs into larger organisations. This approach goes in the way of world-wide manufacturing approaches such as described in the IMS initiative (Wright, 1995). Our work can also apply to a variety of situations, from production line automation to remote site supervision, and many teams are presently working on these aspects, be it at machine, cell, workshop or factory level (Lutz 1999), (Aguirre 1998), (Büllinger 1998), (Cheng 1999), (Fernandez 1999).

In the following, the main modelling features of MMS and OSACA are first recalled. The way these architectures have been implemented within the CORBA context are then presented, including a demonstration platform. In the last part of this paper preliminary results in the field of CORBA with real-time communication constraints fulfilment are given.

2 The industrial models for distributed manufacturing control

2.1 An object-oriented interpretation of the MMS model.

MMS (*Manufacturing Message Specification*) is primarily an application layer of the ISO/OSI protocol stack. Its objective is the remote control or monitoring of any computerised and programmable manufacturing devices such as: machine tool, robot or automated equipment controllers. Together with a wide-band Token Bus physical layer, it constitutes MMS-MAP, the *Manufacturing Automation Protocol* (Valenzano 1992). MAP/MMS is an expensive solution to install, tune and maintain, and it arrived at a time when legacy control systems were vendor specific and difficult to adapt. However a lot of work was done by both the IT and manufacturing experts to define the MMS layer. We believe the MMS work is worth being considered by itself, even if our proposed implementation departs from the orthodox ISO/OSI stack.

In the original MMS environments *Virtual Manufacturing Devices (VMD)* associated with the real Devices and NC Controllers provide *Services* to remote *Users* or clients. Communication is essentially based on a negotiated *Association* and takes place through exchanged messages or *Protocol Data Units* (figure 1) with standardised formats specified in the *ASN.1* abstract syntax. Co-operation is usually implemented in the form of confirmed (positive, negative or error) services, initiated by a remote *User*. A few unconfirmed services initiated by a Server messages do exist (e.g. *Notification* about a VMD status change).

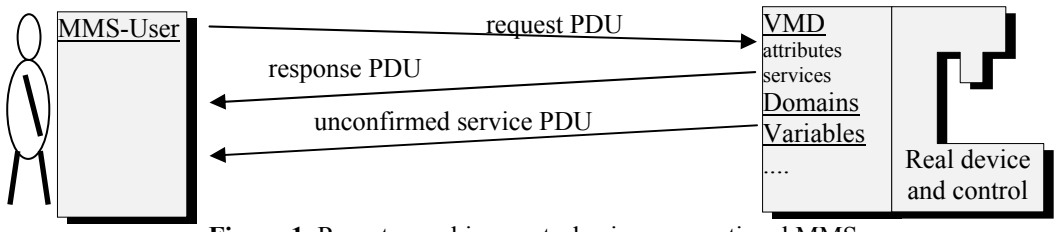

Figure 1. Remote machine control using conventional MMS

The VMD is the real system abstraction visible from a distance. It is an object that encapsulates attributes (identity, status,..) and methods or *services*. The *MMS Client Service Provider* lets a *MMS User* query information or request services from the VMD and objects within the VMD. Some sixteen object classes, or abstractions, are defined within a VMD. The best known and most useful are:

- *Domains*, which are loadable sets of user resources (data and code) ;
- *Program Invocations* which are the execution of user programs (a machining program, but also a return-to-origin program, or a logic controller program managing an auxiliary task) ;
- *Variables* and Sets of variables of VMD or Domain scope ;
- *Event* related objects: conditions enrolments and actions.

The MMS standard (ISO 1990), and the so-called *Companion Standards* relative to NC Machine Tools (ISO 1991), Robot Controllers, Programmable Controllers and Process Controllers give important guidelines for mapping the very general MMS model to each of these particular equipment classes.

The neat presentation of the MMS model as a collection of entities encapsulating a*ttributes* and s*ervices* (methods) allows considering it as an *Object Based* model, using (Wegner 1987) terminology, but neither a *Class Oriented* model (no instantiation, although this can be discussed), nor an *Object Oriented* model (no inheritance). The first point however is enough for considering object-oriented technologies for implementing MMS abstractions and services.

2.2 The OSACA architecture and approach to object oriented communication.

Beside external openness, there is a demand from OEMs and final users for easier replacement of one logical component for another or adding a new functionality in an industrial controller. While the first aspect mainly concerns communication means and protocols, the second concerns *control architectures* (Pritschow et Al. 1996) (Pritschow et Al. 1997) (Raddadi et Al. 1995).

The MMS standard considers the externally visible features of manufacturing devices. It is the aim of the so called MMS *NC Companion Standard* (NC-CS) to map real world Numerical Controller parts into MMS abstractions. NC-CS defines twelve basic *domains* (Device, Process, Process Information, Tool Data, Set-up Data, Device Origin Set-up...) which correspond to effective logical or physical components. NC-CS also defines more in depth *Program Invocations* and *Named Variables*, not to speak of a non limited list of additional structured objects such as *auxiliary tool magazines* or *handling and parking systems.* Still MMS's scope remains essentially remote supervision.

More recent work have resulted in significantly improved agreement today's and future NC architecture. The European Esprit III OSACA initiative (Open System Architecture for Controls within Automation Systems) has brought together academics and industrials from 1992 to 1997 to tackle the standardisation of the internal control architecture modules (OSACA 1997). The main goal of OSACA was to define a hardware and vendor independent reference architecture for numerical controllers that will facilitate the work of NC integrators and provide the users with enhanced possibilities of incremental improvement of their equipment (Lutz and Sperling 1998).

OSACA reference architecture is first described in terms of *Functional Units* (FU), such as those presented in the following Table 1.

Functional units (FU)	Description
Motion Control Manager (MCM)	mandatory when several simultaneous motions must be co-ordinated
Motion Controls (MC)	deals with program decoding, interpolation, tool correction,..
Spindle Controls (SC)	controls spindle rotation, and possible gear change,...
Axis Controls (AC)	the servo-interfaces to one or several axes
Co-ordinator of Unit (CU)	a controller to synchronise several NC-systems in a cell
Logic Controller (LC)	the usual logic control of all alarms and auxiliaries in an NC-system

Table 1. Functional decomposition in OSACA.

The implementation of these functions is through the encapsulation one or more FU's into *Architecture Objects* (AO), autonomous entities which include *Communications Objects* (CO). Of practical importance is the definition of a *Configuration System,* based on particular Architecture Objects, which manages the initial dynamic loading and instantiation of those executive Architecture Objects that are needed.

Internal communication is a fundamental feature of the OSACA reference architecture. All components have well defined interfaces and an internal OSACA *Message Transport System* (MTS) has been defined for inter-AO communication, it is message oriented. However the OSACA communication system belongs to a pre-ORB generation, it is architecture specific.

Our project investigates how a CORBA communication approach can be extended to inter-AO communication and support OSACA objectives. It will ultimately require real-time efficient Object Request Brokers that can recognise local invocations and bypass the lower usual transport layers. Conforming as much as possible to OSACA's system analysis will simplify common understanding.

As a demonstration platform, a PC and NC Board based controller will in fact consist in one *Motion Control Manager*, one *Motion Control*, and one or several *Man Machine Interfaces* (MMI). The MMI of the local system will be very similar except for a few switches and push buttons, which are the privilege of the local operator (such as hand-wheels, or switches). A strategic question is: should the MMI use MMS or OSACA's Application Service Systems ? An MMS-Client could indeed be locally installed and be managed by the MMI, meanwhile the OSACA project promotes decentralisation of the MMI over a local network. For the sake of demonstration, we presently have chosen to use the MMS concepts to implement communication between remote human supervision and the local NC controller.

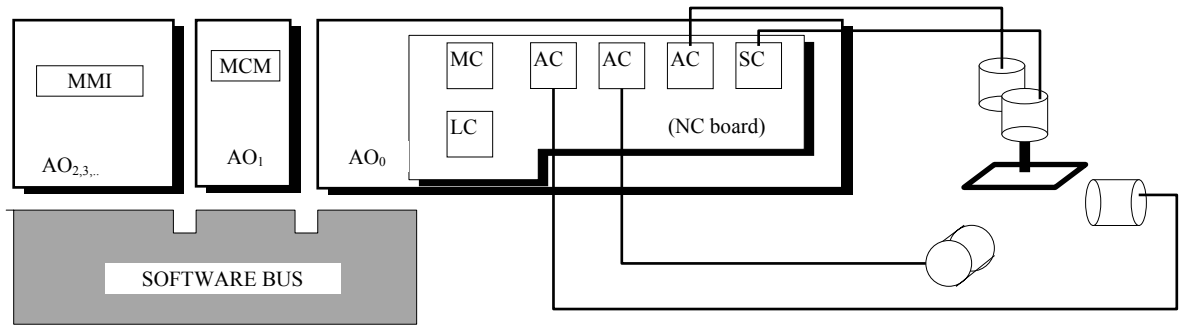

Figure 2. Proposed architecture for NC system.

3 Implementing the reference architectures within the CORBA context

Compared to the heavy and old MMS-ISO solution and the specific OSACA communication architecture, we believe the choice of a standardised object-oriented approach is wiser in the long term. The first emerging and widely accepted standard in this domain was the *Common Object Request Broker Architecture* (CORBA) supported by the OMG consortium.

3.1 The CORBA environment for distributed objects

OMG-CORBA version 2.0 and currently 2.3 (OMG 1999b), is an open architecture standard for distributed objects support and distributed applications. It is freely available for evaluation from various vendors. It brings interoperability, whatever the programming language (e.g. C, C++, Java,..), the operating system (MS-Windows, Unix,..) and the ORB source. The OMG-CORBA standard is the basis of our object oriented communication environment. In comparison Java's Remote Method Invocation (RMI) mechanism is confined to the Java execution environment, while Microsoft's DCOM architecture remains proprietary and relatively OS specific (SISCO 1998).

In the CORBA context, clients view *distributed objects* defined by their *interfaces*. An interface describes the methods implemented by an object. Interfaces are specified using the neutral OMG's *Interface Definition Language* (IDL). A client performs a method invocation towards an object and gets a response. Clients are not aware of the local or remote location of objects. Objects can be clients of other objects. Method invocations behave like TCP-IP based *Remote Procedure Calls* (RPC) but are different. RPC are procedures offered by a dedicated server, while methods are attached to an object and a server can handle many objects (Gressier et Al. 1996).

The OMG's object management architecture is composed of four main elements (Siegel 1996):

- The *Object Request Broker* (ORB), also called the object bus, is the heart of the communication system. It supports interactions between client applications and server objects. Method invocations are translated into requests and responses within the ORB. The ORB hides the heterogeneity of hardware and operating systems.

- *Common Object Services* enhance the basic services offered by the ORB. They can be seen as complementary system components. There are many Common Object Services, *object naming* and *life cycle*, are the most frequently used. These services are implemented as co-operating objects with a well known interface specified using IDL.
- *Common Facilities* define general purpose or domain specific collections of components that can be used by application objects.
- Application *server objects* are specifically those developed by users for their applications.

The current 2.3 version, enhances the portability of server objects, and provides some specifications to provide the interoperability of CORBA with other distributed architectures such as DCOM. Furthermore, the future versions should address issues related to message oriented communications, real-time, quality of service, and fault tolerance.

DCOM, a Microsoft based alternative to CORBA has emerged from the evolution of the Windows OLE/COM mechanisms for inter-object communication. The OPC (OLE for Process Control) consortium (OPC 1999) followed this evolution and disseminated this mechanism in the area of process control, data acquisition and more recently of Numerical Control and factory automation (Allain 1999). This also seems the choice of the OSACA/HÜMNOS consortium (Lutz 1999). All these recent developments or declarations comfort choices we had taken a long time ago (Razafindramary et Al. 1996).

3.2 Implementing MMS in the CORBA environment

A distributed object-oriented version of MMS has been designed at CEDRIC, assuming the underlying ORB is CORBA 2.0 compliant (Guyonnet 1997),(Gressier 1999). It has been derived from the ISO-MMS standards mainly by adapting the following features.

The generic adaptation to an object oriented model:
VMD are promoted to primary *server objects* that can be manipulated directly by the applications and related services are uniformly handled as generic interfaces. Encapsulation is enforced, and as a consequence VMD internal resources (such as variables, domains, program invocations) can be handled only through specific methods supported by VMD interfaces. These abstractions still exist but only as VMD configuration parameters (these abstractions are implemented with the selected programming language, and do not possess any dedicated remote interface).

The specific adaptation to the CORBA object model:
The ISO message oriented communication scheme has been replaced by standard CORBA interactions between client and server objects. The overall architecture is described in figure 3. More precisely:

- confirmed services (for example *Identify*): They are mapped onto standard CORBA (synchronous) method invocations. A non-blocking behaviour of the calling object can be obtained if the method invocation is performed in a separate thread (when available on the underlying platform) ;
- unconfirmed services (for example *InformationReport*): They were initially mapped onto *one-way* method invocations , but no reception guarantee would have been offered. They thus have been mapped onto a *void* standard method invocation. A non-blocking behaviour can be obtained in the caller by *multithreading*.

Objects have been given *names* and can be accessed in the system through *object references* (in ISO-MMS, entities manipulated in the program have no name and can only be designated through *associations*). ASN.1 message specifications have been mapped to IDL interface definitions. MMS request PDUs are transformed in method parameters. MMS positive response PDUs are transformed in method results. Error PDUs are transformed into CORBA exceptions.

To be closer to the services provided by the ISO-MMS standard, three additional CORBA common object services could be used: *Persistency*, *Events* and *Concurrency*. ISO-MMS deals with persistency through Domain upload and download, persistency brings a more efficient way to store domains when they are no more in use. The replacement of the ISO-MMS Event Service by the CORBA Event Service should be evaluated, because they behave differently. ISO-MMS also offers *semaphores*, and it should be investigated how the CORBA Concurrency service could map the requirements of the MMS semaphores.
Moreover, we could experiment the benefits of a Trading service. One can consider the wide use of Internet for business in the field of manufacturing. A salesman could select the right provider to get a product via the trading service where different companies able to deliver it are registered.

Implementations of these principles have been developed at CEDRIC over a number of operation systems and Object Request Brokers:

- MMS for C++ application programs, using Chorus COOL ORB (Chorus 1997) and Chorus micro-kernel OS (Guyonnet et Al. 1997) ;
- MMS for Java application programs, using ORBacus (OOC 1998), Sun Microsystems' JDK Developer's Kit and Windows 95 or Linux (Laurent 1998). The use of Java in our last prototype brings code portability. Today kernels generally support a Java Virtual Machine. Code portability allows quick program migration and eases reconfiguration in case of failure.

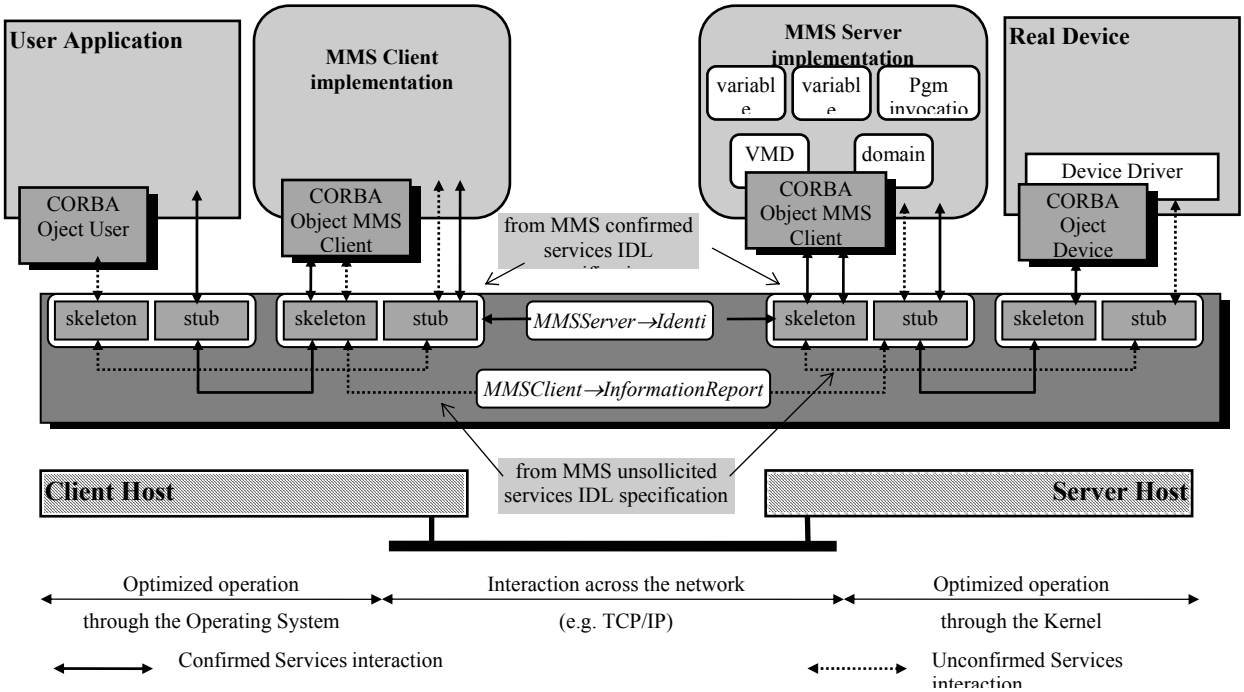

Figure 3. Principles of MMS over a CORBA software bus

3.3 A demonstration prototype

After the preliminary works which proved off line feasibility of the object-oriented version of MMS, a more demonstrative project was decided, consisting in the remote control of an effective machine-tool through our MMS for Java/CORBA (Laurent 1999) prototype.

Services and structures specified in ASN1 have been translated automatically into IDL using an ASN1 to IDL compiler used by (Mermod and Genilloud 1998) and described in (Orbycom 1998). The results can be lightened from those services that are taken in charge by CORBA general services.

The following MMS abstractions with their most meaningful services have been implemented:

- VMD (access to attributes, list of objects or named variables)
- variables (read / write significant variables)
- domains (download a program),
- program invocations (start / suspend / resume / stop a machining program or set up process)

The machine-tool is a legacy 3-axes Wirth & Gruffah milling machine with renovated DC motors. The control system is a DeltaTau PMAC© 4-axes DSP processor board installed in a PC (DeltaTau 1995). This NC board gathers sufficient intelligence to be considered as a full NC Controller with capabilities for machining program execution and PLC driven auxiliary functions. The host PC works under the Linux operating system, it will run the MMS Server object. A Linux driver for the PMAC board has been developed ; it is written in C and accessed by Java programs through the Java Native Interface (Epivent 1998). The controller PC, by the machine, can communicate through an Ethernet LAN with an MMS NC User installed on another PC, which is running under Windows'95.

Although the platform is voluntarily non-homogeneous, a common development environment has been installed: Object Request Broker ORBacus (OOC 1998) has been installed on both machines, as well as Sun's Java Development Kit version 1.1. The choice of Java programming allows a relatively fast development and a perfect independence regarding the OS since Java applications run inside the uniform Java Virtual Machine environment.

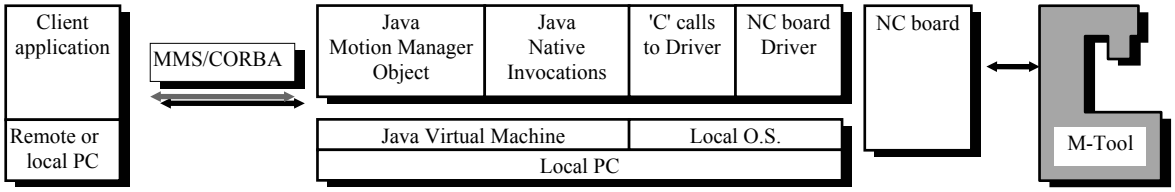

Figure 4. . Architecture of Java/CORBA MMS demonstrator.

The Motion Manager cares for a simplified local state machine (Manual/Auto modes), it is included in the VMD.

The demonstration scenario consists in:
- spawning the CORBA naming service and resetting the Client and Server objects ;
- requesting VMD information ;
- downloading a machining program (Domain Download service) ;
- starting, suspending, resuming program execution (Program Invocation service).

The overhead due to the MMS for Java/CORBA interface was tested equal to 3 ms per confirmed service invocation on one 200 MHz Pentium. Remote tests on an Ethernet LAN do not add significant overhead. However IP and Ethernet do not guarantee any quality of service (QoS). Great improvements in reliability and response times are expected from future experimentation with real-time ORBs.

4 Addressing Real-Time Communication Constraints

4.1 Users Requirements

The preceding paragraph has shown the feasibility of using the MMS concepts within the CORBA distributed object context. MMS was originally designed for relatively slow interactions in which the NC User basically deals with control-command interactions with NC devices. Video monitoring of the manufacturing process was actually historically a concern of the architects of MAP-MMS. For that purpose, the MAP Physical Layer was a video coaxial cable with add-on wide-band data support. Present evolution is to include audio and video streams on a unique digital data channel (Sempere 1999). In case of long distance control, through Internet for example (Razafindramary et Al. 1996), audio streams can carry information for fault detection on devices, one can think about old workers and golden ear expertise. Video streams can be helpful for operation control or final product control.

As a consequence, so called *Quality of Service* (QoS) requirements are expressed in terms of:
- *throughput* expressed in messages per second or in transactions per second,
- *transfer latency*, response time at the communication level or at the application level expressed in milliseconds,
- maximum *jitter* or transit time fluctuation,
- admitted message *loss ratio* (meaningful only for audio and video).

Based on our experience gained with the work described in the previous paragraphs, we decided to tackle the integration of these real-time constraints in our existing MMS for Java/CORBA prototype. The next paragraphs reports on this project (Gressier-Soudan 1999).

4.2 The Flexible ORB approach: Bindings to handle QoS requirements

Instead of the ORBacus ORB used for the original MMS for Java/CORBA prototype, we decided to switch to a flexible ORB called Jonathan (Dumant et Al. 1998). The architecture of Jonathan allows to plug new types of communication protocols and networks within the ORB. With this feature, the ORB can support real-time communications. Due to the conformance of both ORBs to the CORBA specification, this switch lead to very little

changes in the code. Jonathan's main benefit is not the ORB itself but its conformance to *the Open Distributed Processing Reference Model* (RM-ODP) and its extension dedicated to QoS management for multimedia applications called ReTINA (Blair and Stefani 1997). This model offers a framework that enables QoS requirement specifications and computation (Février and Al. 1997) (Leboucher and Najm 1997). We believe this is a key property to address real-time constraints in manufacturing application design.

RM-ODP introduces the concept of *binding*. Two objects may interact directly (through method invocations in the same execution context). Also, they can invoke the same method on a binding object whose role is to transmit the invocation and the corresponding result. Binding objects are usually composite distributed objects that encapsulate the full end-to-end computational resources between interacting objects. When temporal guarantees are involved, a binding object generally encapsulates:

- a specific networking infrastructure (i.e. prioritised Ethernet VLAN IEEE 801.2.1p, or an ATM or FDDI network);
- a specific transport protocol, specific stubs and skeletons (using specific encoding/decoding algorithms, specific buffer and thread management policies);
- a specific object adapter (implementing specific thread dispatching/scheduling policies), etc.

In this model (depicted in figure 5) each binding has a type that represents the protocols used, possible quality of service constraints (throughput, latency…), or any other kind of binding property (Seinturier et Al. 1999). Bindings are generated using binding templates provided by *binding factories*.

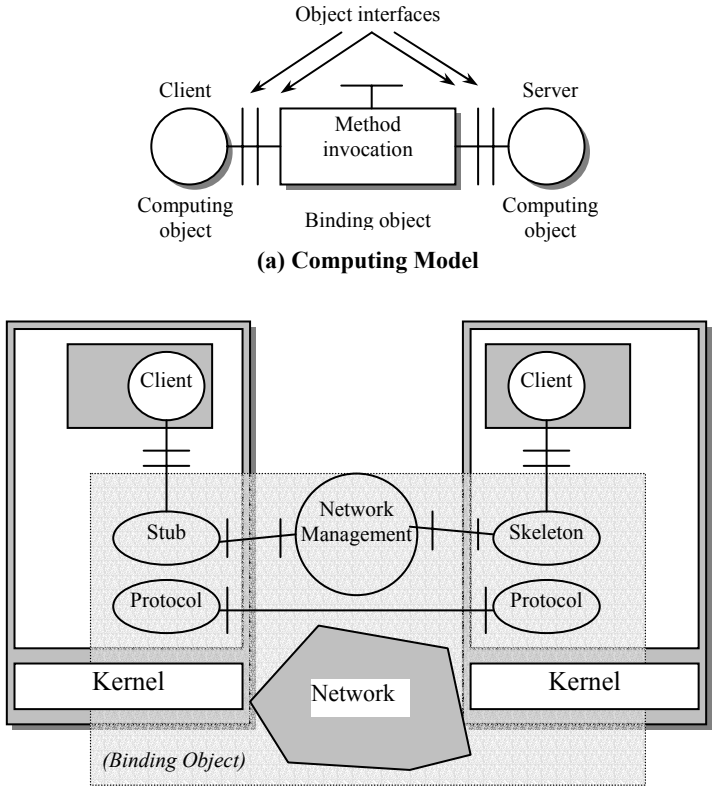

Figure 5. Reference model for Open Distributed Processing and Binding objects

The prime characteristic of binding based architecture is the ability to plug arbitrary forms of binding policies. This goal has been achieved by designing a minimal ORB kernel whose role is to provide a generic environment that can be used by any binding factory to create and to manage specific bindings. With this architecture, it is possible to introduce new communication mechanisms (protocols, services, resource management policies) and invocation semantics encapsulated in binding factories written by application developers and possibly discovered and installed at run-time. This design conforms to the pluggable protocol paradigm implemented in TAO (O'Ryan et Al. 2000). The resulting minimal ORB offers maximum flexibility. Currently, Jonathan comes with two personalities: a CORBA 2.0 ORB (David), and, Java RMI (Jérémie) (France-Telecom 1999). With the binding framework, such personalities could be based on a fieldbus network instead of LAN based TCP/IP network.

4.3 Preliminary prototyping with the AJAX toolkit

We enhanced the MMS for Java/CORBA prototype with real-time communications based on the ATM technology, which is a good candidate to handle real-time communications. It also provides the ability to handle simultaneously data and audio/video traffic. The prototype uses the AJAX toolkit within the Jonathan ORB. AJAX is a binding factory (Blair and Stefani 1997) (Dumant et Al. 1998) that can be used to set up ATM connections for CORBA objects (Seinturier et Al. 1999). This prototype uses the ATM library on Linux (Almesberger 1997). The underlying transport protocol is AAL5. This last proposal offered an object oriented QoS aware ATM based industrial messaging service.

Our testbed platform for AJAX was composed of two 200 MHz Pentium Pro PCs with 155 Mbps ATM adapters from Efficient Network Inc. connected by a switch. They ran a Linux 2.1.x kernel with a JDK 1.1.x environment from Sun. First performance tests showed that the ATM CBR service class (Constant Bit Rate) performed well for requested throughputs of up to 10 Mbps between pairs of MMS objects. Ten bindings were able to be open simultaneously on the testbed platform.

Unfortunately, some problems arise with requested throughputs greater than 10 Mbps. They are mainly due to the deep nature of the CBR service class and of the AAL5 protocol layer. Indeed, CBR provides a constant bit rate but AAL5 does not guarantee the delivery of data. This leads to the situation where data may be lost if for instance, the receiving side does not read the data as fast as needed. Performance tests (Almesberger 1997), conducted by the authors of the ATM on Linux library, show that a requested throughput of 130 Mbps between two PCs, connected back to back, leads to 0 to 13 % loss of data with a simple client/server program written in C. Our own tests show that this loss ratio is reached for requested throughput of 80 Mbps when the PCs are connected through a switch. Unfortunately, when the client/server program is written in Java, this loss ratio is reached as soon as 15 Mbps.

Despite these weak results, we believe that our work is promising to address real time constraints. We have analysed the previous results. Two main causes justify the loss ratio in the previous results: first, the Java virtual machine introduces an overhead in program executions and second, the two distinct memory zones (one for the virtual machine and one for the C native code) of the Java Native Interface (JNI) introduces an additional copy of each received packet. In our project, an experiment with ATM, an optimised Chorus OS micro-kernel and a JVM based on Japhar (Reinholdtsen 1998) obtain better results (Lizzi et Al. 1999). Communications between Java threads on two different hosts (no JIT, optimised JNI buffer management, zero-copy buffers) deliver a throughput of 97 Mbps, an end-to-end delay of 15 ms (1 switch crossed), with a loss ratio of 0.3 % (due to buffering losses, no cell losses). The micro-kernel implements a zero-copy buffer mechanism to handle messages, Japhar's Java threads are mapped on real-time micro-kernel threads, JNI doesn't make any additional copy, and real-time port based inter-process Chorus communications replace socket based message exchanges. The design of Lizzi's platform is efficient but too specific to be of current use. One of the improvements that can be considered to tackle our problem of loss ratio in our prototype is to use a real-time kernel and a more efficient Java Virtual Machine that better deals with real-time performances as, for example, in PERC (Nilsen 1996).

5 Conclusion

This paper has presented our efforts to keep the fundamentals of the prevailing standards while looking for solutions which are as easy and universal as possible. We have proposed and applied a methodology for implementing an object-oriented industrial messaging service.

The guidelines of this work have currently been presented and acknowledged at the OMG Manufacturing Domain Task Force (Gressier-Soudan 2000). The adopted protocol approach fits the requirements of the Extensible Transport Framework for real-time CORBA RFP (OMG, 2000-a). Our results could be a contribution to the definition of real-time messaging services in the context of OMG's Embedded Systems workgroup related to Smart Transducers (OMG, 2000 b).

In automated production contexts, real-time remote control may become the main factor and one will ask for real-time properties in software buses. Our latest realisations concerned real time constraint fulfilment for remote control using an ATM network. For short scale or embedded distributed control fieldbus support could be preferred, the binding object approach is an appropriate method for accommodating any networking infrastructure.

Large NC controller's manufacturers like Siemens AG or Num/Schneider SA have shown their awareness by recently committing to the OPC architecture based on Microsoft's DCOM approach. Meanwhile work groups are

at work at the OMG Consortium and at the *OLE for Process Control* Foundation to find gateways between these similar and concurrent communication architectures.

Going into the world of CORBA communication opens new horizons for local and also for long distance monitoring or diagnosis of devices. Software buses were expected to be primarily used for business applications. The same could be said for Ethernet, which now also finds niches in factory applications. These tools allow filling the gap between office and factory. Business application are particularly interesting in the production area when one considers small production entities with human operated NC Systems: there will be a need for on line technical decisional help on the manufacturing stations.

References

AGUIRRE SUAREZ O., AJURIA FORONDA J.L. AND MARTIN ABREU F., 1998, Standard based framework for the development of manufacturing control systems. *International Journal of Computer Integrated Manufacturing,* **11**(5), 401-415.

ALMESBERGER, W., 1997, ATM on Linux - the 3rd year, *4th International Linux Kongress*, March.

AMICE-CIMOSA CONSORTIUM, 1993, *Open System Architecture for CIM* (Berlin Springer Verlag)

ALLAIN M., 1999, AXIUM: le dessous des cartes. *NUM Information*, nr 32, 6,November.

Blair, G., Stefani, J-B., 1997. *Open Distributed Processing and Multimedia*. Addison-Wesley.

BULLINGER H.J., FAHRICH K.P., LISENMAIER T., 1998, A conceptual approach for an architecture of distributed objects for the integration of heterogeneous data processing systems in manufacturing companies*. International Journal of Production Resarch,* **36**(11), 2997-3011.

CHENG F.T., SHEN E., DENG J.Y., NGUYEN K., 1999, Development of a system framework for the computer-integrated manufacturing execution systems: a distributed object-oriented approach. *International Journal of Computer Aided Manufacturing,* **12**(5), 384-402.

CHORUS SYSTEMS, 1996, Chorus/COOL ORB Programmer's Guide. CS/TR-96-2.1.1996, Paris.

DeltaTau, 1995, *PMAC Lite Developper's Handbook,* (http://www.deltatau.com, DeltaTau Inc.)

DUMANT, B., DANG TRAN, F., HORN, F. AND STÉFANI, J.-B., 1998. Jonathan: an open distributed processing environment in Java. *Proceedings of Middleware'98, IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing.* The Lake District, UK, September.

EPIVENT, M., 1998, Driver sous Linux pour Carte de Commande Numérique. Final work, CNAM/CEDRIC, September 1998, Paris.

FERNANDEZ, J.A., GONZALEZ J., 1999, The NEXUS open system for integrating robotic software. Robotics and Computer Integrated Manufacturing, **15**(6), 431-440.

FEVRIER, A.,NAJM, E., STEFANI, J-B,1997, Contracts for ODP. *Fourth AMAST Workshop on Real-Time Systems, Concurrent, and Distributed Software,* Mallorca, May 21-23 (Springer-Verlag).

FRANCE-TELECOM/CENT, 1999. Jonathan An Open Distributed Processing Environment, http://www.objectweb.org/en/jonathan.

GRESSIER-SOUDAN, E., LEFEBVRE, M.AND ROZIER, M., 1996, Protocole de messagerie MMS sur CHORUS : différentes approches. *Proceeding of the RTS'96 Real Time Systems Conference,* Paris, January.

GRESSIER-SOUDAN, E., SEINTURIER, L., 1999, Object Oriented Industrial Messaging Services : Which Approach to address real time constraints ?, Cedric Research Report RR 99-16. Paris. France. December 1999.

GRESSIER-SOUDAN, E., 2000, Prototyping a CORBA based MMS-Industrial communications , OMG TC Meeting, Manufacturing Domain Task Force. Burlingame. USA. September 2000.

GRIMES, R. 1997, *Professional DCOM Programming*. (Birmingham: Wrox Press).

GUYONNET, G., GRESSIER-SOUDAN, E. AND WEIS F., 1997, COOL-MMS: a CORBA approach to ISO-MMS. *Proceedings of ECOOP'97, Europeean Conference on Object-Oriented Programming,. Workshop on CORBA Implementation, Use and Evaluation*. Jyväskylä, Finland, June.

ISO, 1990, *ISO 9506-1. Industrial Automation Systems - Manufacturing Message Specification - Part 1: Service Definition.* (Geneva: International Standard Organisation).

ISO, 1991, *ISO 9506-4. Industrial Automation Systems - Manufacturing Message Specification - Part 4: Companion Standard for Numerical Control.* (Geneva: International Standard Organisation).

LAUKIEN, M. AND SEIMET U., 1997, OmniBroker Version 2.0.2. Object-Oriented Concepts Inc.

LAURENT, A., 1998, OO-MMS: une approche CORBA/Java pour ISO-MMS. Systèmes de Communications Industrielles, CEEA Seminar, http://www.rli.cran.u-nancy.fr/sci, Nancy, 6-7 July.

LAURENT, A., 1999, Une approche CORBA/Java pour la commande numérique. Final diploma dissertation. CEDRIC/CNAM, Paris, June.

LEBOUCHER, L., NAJM, E., 1997, A framework for real-time QoS in distributed systems,. *IEEE Workshop on Middleware for Distributed Real-Time Systems and Service*, San Francisco, December 2.

LIZZI C., GRESSIER-SOUDAN E. AND MONTIEL J., 1998, A real-rime communication service for ATM-based distributed systems. Proceedings of the IEEE Conference on ATM, Colmar, France, June.

LIZZI, C., 1999. Java Real-Time Distributed Processing over ATM Networks with ChorusOS*, Proceedings of the ETFA'99 Conference on Emerging Technologies for Factory Automation*, Barcelona, October.

LUTZ, P.AND SPERLING W., 1997, OSACA, the vendor neutral control architecture. Proceeding of the Iim'97, Internatioanl Conference, Dresden.

LUTZ P., 1999, Wie wichtig ist ein bestellerübergreifende offen Steuerungarchitektur in PC Zeitalter ? *Fertigung,* **(09)**12 (Verlag Moderne Industrie AG, Landsburg).

MERMOD, A.AND GENILLOUD, G., 1998, MMS over CORBA. Private communication. February 1998.

NILSEN K., 1998, Issues in the design and implementation of real-time Java. Technical Report NewMonics Inc. http://www.newmonics.com.

OMG, 1999a, *CORBA Beginners page*, http://www.omg.org/corba/beginners.html.

OMG, 1999b. *CORBA/IIOP 2.3.1 Specification. Document ad/99-10-07 1999,* Object Management Group, http://www.omg.org.

OMG, 2000a, Realtime Notification RFP. OMG document orbos/00-06-10.

OMG, 2000b, Smart Transducer Interface RFP. OMG document realti/00-07-01.

OPC, 1999, Ole for Process Control consortium, http://www.opcfoundation.org.

OOC, 1999, *ORBacus presentation*. http://www.ooc.com/ob, (Billerica (USA), Ettlingen (D): Object Oriented Concepts)

O'RYAN, C., KUHNS, F., SCHMIDT, D.C., OTHMAN, O., PARSONS, J, 2000, The Design and Performance of a Pluggable Protocols Framework for Real-time Distributed Object Computing Middleware, submitted to *IFIP/ACM Middleware 2000 Conference*, Pallisades, New York, April 3-7.

OSACA, 1997, *OSACA 1.0.1 Handbook deliverable*, http://www.osaca.org, (Bern (CH): Osaca Consortium).

PRITSCHOW G., DANIEL C., JUNGHANS G. AND SPERLING W., 1994, Open System Controllers, a challenge for the future of the machine tool industry. *CIRP Annals*.

PRITSCHOW, G. AND SPERLING, W., 1996, Information interchange in open control systems. *Wissenschaftliche Gesellshaft für Produktionstechnik*, Darmstadt.

RADDADI, M., RAZAFINDRAMARY, D., BOISSIER, R., FOFANA, M. AND SORIANO T, 1995, Modeling and design of machine-tool controllers for small CIM units. *Proceedings of the ETFA'95 Conference on Emerging Technologies for Factory Automation,* Paris, October.

RAZAFINDRAMARY, D., FOFANA, M., GRESSIER-SOUDAN, E., RADDADI, M., BOISSIER, R. AND PONSONNET R., 1996, Architecture Réceptacle et services pour l'intégration CFAO-Usinage dans les petites entreprises. *Proceedings of IDMME'96 Conference on Integrated Design and Manufacturing in Mechanical Engineering*, Nantes (F), April. (Dordrecht: Kluwer).

REINHOLDTSEN, P., 1998, Japhar : the Hungry Programmers Open Source Java. Norwegian Unix Users Group. November.

RODD M.G. AND ZHAO G.F., 1990, RTMMS: an OSI-based Real-Time Messaging System, *Journal of Real-Time Systems*, **2**.

SEINTURIER L., LAURENT A., DUMANT B., GRESSIER-SOUDAN E. AND HORN F., 1999. A framework for Real-Time communication based Object-Oriented industrial messaging services. *Proceedings of the ETFA'99 Conference on Emerging Technologies for Factory Automation,* Barcelona, October 1999.

SIEGEL, J., 1996, *CORBA Fundamentals and Programming*. John Wiley.

SEMPERE V.M., MATAIX J., TORDERA E., 1999. Video transmission on industrial processes over MAP networks using MMS. *Proceedings of the ETFA'99 Conference on Emerging Technologies for Factory Automation,* Barcelona, October 1999.

SISCO, 1998. ActiveMMS offers enhanced productivity. *Host Newsletter* **9**(3), http://www.sisconet.com.

SUN, 1998, *Java Remote Method Invocation,* Sun Microsystems, http://www.javasoft.com/jdk/rmi/.

ORBYCOM, 1998, GIDL: a GDMO ASN1 to IDL translator, Orbycom, http://www.orbycom.fr/gidl.html Fontainebleau (F).

VALENZANO A., DEMARTINI C. AND CIMINIERA L., 1992, *MAP and TOP Standards and Applications*, Addison Wesley.

P. WEGNER, 1987, Dimension of object-based language design. *Proceedings of the ACM Conference on O-O Programming,* 168-182.

P.K. WRIGHT, 1995, Principle of open architecture manufacturing. *Journal of Manufacturing Systems*, **14**(3).