ABCM Symposium Series in Mechatronics - Vol. 1 - pp.580-589

DISTRIBUTED OBJECT TECHNOLOGIES IN MANUFACTURING EXECUTION SYSTEMS

Paulo M. P. A. Blanco
University of São Paulo – Polytechnic School – Department of Mechatronics and of Mechanical Systems Engineering
Av. Prof. Mello Moraes, 2231 – São Paulo – SP – 05508-900 - Brazil
paulo.blanco@eds.com

Marco A. Poli
University of São Paulo – Polytechnic School – Department of Mechatronics and of Mechanical Systems Engineering
Av. Prof. Mello Moraes, 2231 – São Paulo – SP – 05508-900 - Brazil
marco.poli@poli.usp.br

Marcos R. Pereira Barretto
University of São Paulo – Polytechnic School – Department of Mechatronics and of Mechanical Systems Engineering
Av. Prof. Mello Moraes, 2231 – São Paulo – SP – 05508-900 - Brazil
mrpbarre@usp.br

Abstract. Integrated manufacturing systems have always found in the hardware and software heterogeneity among different vendors an obstacle to their satisfactory implementation. Nowadays, new Open and Distributed Systems technologies have appeared, providing a new integration potential, accelerating the implementation of MES (Manufacturing Execution Systems) and making it's integration to other corporate systems simpler.
This work is intended to review the technology involved in distributed objects used in MES, comparing the standards and the available software resources. The paper will first discuss the new possibilities for MES implementation opened by distributed objects: a new generation of pro-active PLCs (Programmable Logic Controllers), intelligent sensors and distributed plant management. Then, it will review OPC/DCOM and CORBA, as the prevailing technologies for distributed MES implementation. These two technologies will then be compared when aiming to implement MES. Finally, an industry survey will be presented showing which technology is being used by some of the topmost MES currently commercially available.

Keywords*. Distributed Objects, Manufacturing Execution Systems, OLE for Process Control, Common Object Request Broker Architecture, Commercial MES.*

1. Introduction

Since the last decade, many industries around the world implemented their ERPs (*Enterprise Resource Planning*) . The ERP made by companies like PeopleSoft, Baan, JD Edwards and SAP improved the integration between applications combining common business processes in one only system, including finances, material management and production planning. (Mello, 2002 and Brown, 2000) The ERP packages are commonly used with business reengineering techniques to update and make better management of a number of support systems in the company. (Brown, 2000)

However, while corporate management systems are naturally turned to long term operations, there is a new and growing tendency that MES can have a considerable impact in short term objectives, as well as in performance and in the capacity of the company to achieve its goals. (OMG, 1997)

The needs for integration between shop floor and business management come from four different sources: the necessity of having better response-time to clients' needs; the flexibilization of the production; cost reduction and quality improvement. (MESA, 1997b) To improve client relations, the factories need to improve quality, costs and delivery timetables under variable conditions. Managers need to be able to coordinate new requests, with new needs, with their existing equipment and existing materials to achieve flexibility.

There is also a growing perception that there is sensible productivity gain when Internet and its technologies are used to make information available in a faster and more efficient way though the company. (Roy et al, 2002)

Examples of problems that can be minimized by the integration of the factory include the management and coordination of production activities, efficient resources allocation, conformance to standards and regulations of specific business markets (like pharmaceuticals) and the reduction of the products' time-cycles. (OMG, 1996)

One of the great challenges of modern industry is the integration of processes and manufacturing information with other processes from the company. Many manufacturing environments consist of a set of applications that manages different activities in the company, and that were conceived in a vertical manner, coexisting side by side, but with no horizontal integration. This way, similar information can be managed in a redundant way that can be incompatible between applications like shop floor control and requests management. Good management of a company requires that information must be shared and flow nicely between business processes. (OMG, 1996)

It is evident that the necessity of a solution for the question of the integration between planning and manufacturing comes directly from the investments made by the industry in the acquisition of corporate systems and the will to lower delivery timings and raise productivity and flexibility. To fulfill this gap, the adequate technology is the *Manufacturing Execution Systems*, or MES.

MES consist in an intermediate layer between shop floor and the corporate management system and are responsible for the integration between, for example, the registration of a new order and its insertion in the production planning of a given machine. (MESA, 2001) Although the basic concept behind MES is known for many years now, the beginning of its application started from three basic reasons: popularization of ERP *(Enterprise Resource Planning)* systems, the recent availability of adequate PC *(Personal Computer)* platforms suitable for supervision and control and the concept of *e-manufacturing.* (Kamal, 1998)

MES products focused in supervisory control integration with the ERPs working on PC platforms and make requests, control and management over the Internet available are already a reality and are available for acquisition as COTS *(Commercial Off the Shelf)* systems, aimed at industrial automation.

Almost all MES packages available today in the market are based on the architecture defined by the products of the American software giant Microsoft. (Hoske, 1998) In fact, the development of the software market for supervisory control itself has been, until today, tied to the availability of solutions based on the Microsoft Windows operating system. Nowadays, the Windows NT system (or its newer versions, Windows 2000 and Windows XP Professional) is the ruling operating system in supervisory control, for its multi-tasking, 32 bits and real-time extensions capabilities. All these characteristics together allow communication in real-time between hardware and software components to be achieved, an essential characteristic for industrial applications.

Together with this fact is the release of the model (or *framework*) developed some years ago by Microsoft for components in distributed applications, known originally as the DNA *(Distributed internet Application)* architecture, that allowed a higher degree of compatibility between components of different suppliers (HOSKE, 1998) and constituted the base for the development of a new, specific technology for data communication in the manufacturing environment, the OPC *(OLE for Process Control)* technology. OPC will be further discussed on a later section.

However, the need to tie the development to one only architecture for implementing MES systems is an illusion. Open architectures are already available, developed by independent organizations that allowed the exact same degree of interoperability and flexibility with the intrinsic advantages of supplier independency. One of this architectures is CORBA *(Common Object Request Broker Architecture),* developed by OMG *(Object Management Group)* and proposes a model fit to the development of interoperability between software objects not independent of the platform on which they are executed, the programming language on which they are implemented and the supplier that developed them.

Combining the potential of this architecture for distributed objects to modern tools like the Java language (platform independent) and the use of PCs on the shop floor, all the necessary infrastructure is available for implementing completely platform independent and operating system MES solutions and do not need to be tied to the PC/Windows environment or computational philosophy but been extended for platforms of bigger or smaller size, according to the needs, adapting itself from big Unix systems to small embedded controllers (as intelligent sensors or pro-active PLCs) running light Java virtual machines. (Highlander, 1998)

2. MES Systems

Manufacturing Execution Systems is a broad concept whose meaning can hardly be resumed in one only sentence. However among the different definitions presented in the available literature one of the most written is supplied by the *Manufacturing Execution Systems Association* (MESA) in MESA, 2001: Manufacturing Execution Systems are systems that supply information allowing for optimization of production activities from the initial request to the final product. Making use of accurate and up-to-date data, MES guide, start, report and answer to activities in plant as they occur. The resulting fast response to operational conditions variations allied to focus in the reduction of activities that do not aggregate value to the product results in the efficient operation of processes in the plant. MES make better return of the operational resources as well as allow an improvement of performance of deadlines, materials management, raw margin and cash flow. MES supply critic information to the company about productivity activities throughout the plant and the supply chain, though bi-directional communication.

As is clear from this definition, the term “manufacturing” from MES isn’t limited to only the productive sectors of a company, but to all its structure, once the concept extrapolates the integration of productive equipment with engineering systems and production management. MES systems, to be efficient, shall make good use also of the information coming from corporate management systems (ERP – *Enterprise Resource Planning*) and from the flow of information present throughout all the sectors of a company.

One of the places where well-implemented MES are useful is in the creation of a bridge between the office and the shop floor, supplying workers in the plant with real, up-to-date data that complement the office-oriented management systems. MES, when applied in conjunction with planning systems make that the work lines and the plan coming from the office be implemented on the shop floor. Basically, the central objective of a MES implementation consists in the reduction of the time taken for the need of change in an industrial process to be changed and the subsequent response-time reduction after a decision is taken.

A localized vision of MES systems relating to the other systems present in a manufacturing corporation can be seen in Fig. (1) used as reference by MESA. In the definition given by MESA, MES operates as an information channel that connects and supplies data to other systems, sometimes having the same function of other kinds of solutions, which in turn can have the same functions between themselves. For example, production programming can be in both MES and SCM *(Supply Chain Management)* systems; resources allocation can be in MES and ERP; documentation can be in

MES and P/PE *(Product and Process Engineering)*. The degrees can vary according to the type of industry and implementation. (MESA, 1997b).

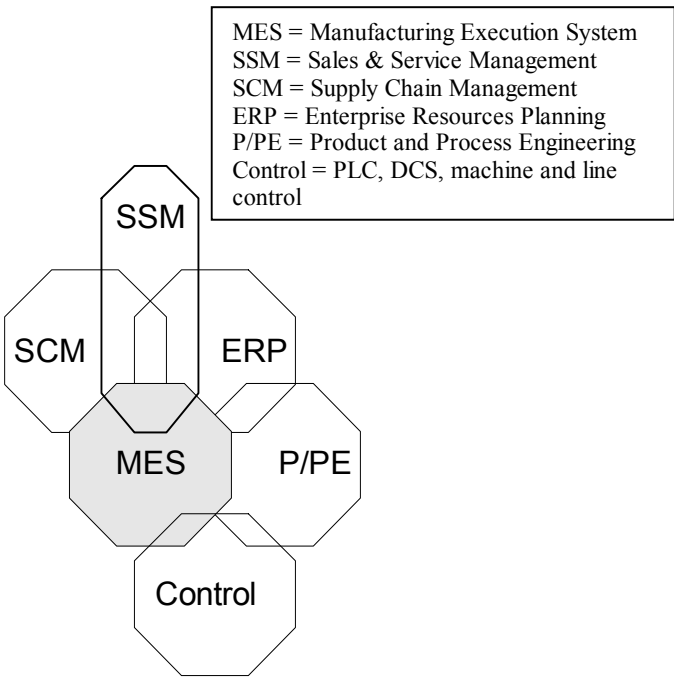

Figure 1: MES context inside corporations (MESA, 1997b)

2.1 MES functions

MES implementations result in the automation of the company's information flow from order request to product delivery, going through all steps in technical and administrative areas. Kamal (1998) says: make information available at the right location, at the right time is the core of integrated manufacturing. A MES system correctly implemented should computationally assist many functions of product manufacturing, from the so called "management systems" to the "production systems" (MESA, 1997b), including:

2.1.1 Resources Allocation and Status

MES should be able to manage production resources available in the plant, such as machines, tools, labor resources, specific abilities, materials and support equipment, and others like documentation that need to be available for production teams to work.

Besides that, MES should be able to provide information about the history of each one of the elements referenced above, make sure that these resources be properly prepared for production activities and track its status in real-time, in a way that production objectives be properly achieved.

2.1.2 Detailed Operations Schedule

MES should develop the sequential operation of the production based on proprieties, attributes, characteristics and revenue associated to each product, in each order. A subset of MES objectives is the reduction of the setup time of machines to the minimum, and to be able to do so, the system should consider, in the scheduling of the activities, aspects like colors, shapes, formulas, analog processes and equivalent components, and to generate the most efficient mixes of operations.

2.1.3 Production Units Allocation

MES should manage the entire product flow in the company, be it by order, final product, work or any other method as desired. The sequence in which these orders are processed should be controlled by MES, taking in consideration the chain of proprieties and all eventual changes in plan that could even occur after the job has already begun. MES should have the capacity of changing the work plan on shop floor and also to consider the use of re-manufacturing and disposal of items during the production process, and to balance the in course activities at all times.

2.1.4 Documentation Control

The production process normally has a set of documentation needed for its adequate operation. MES should be able to make information available to operators like drawings, recipes, diagrams and electrical charts, operational instructions, safety instructions, part lists, production orders and other documents that might be needed for production.

2.1.5 Data Acquisition

MES should generate production information associated to each order and each product, such as production parameters, productivity index, problem reports and alarms generated through the production process. It is desirable that MES be integrated to production equipment in a way that this data is available to operators in real-time, being used for efficient supervision of shop floor activities.

2.1.6 Labor Management

Labor management activities of MES refer to control and allocation of resources, maintenance of up-to-date registers of the specific capabilities of all workers in the production process, controlling their certifications and analyzing their productivity. MES should interface workers' frequency-control systems in such a way that it should be able to relocate labor as necessary.

2.1.7 Quality Management

MES functions related to quality in the productive process refer to the real-time analysis of characteristics of notice of the production items such as *Statistic Process Control* to detect abnormal tendencies in manufacturing and actuate to correct them.

Data collection should be done by specific interfaces connected to the production systems as well as by terminals available on shop floor for operators to type in their data in a periodic manner.

2.1.8 Process Management

MES should monitor the production activities in a continuous way and feed operators with enough information for corrective actions in the processes in case of need. It is advisable that MES identifies and processes the alarms coming from shop floor (received by the data acquisition shown in 2.1.5) in such a way that production managers be aware of situations and have the means to take the necessary actions.

2.1.9 Maintenance Activities Management

The system should make the activities related to planning and scheduling of preventive maintenance, as well as trigger corrective maintenance jobs in alarm situations. A history of maintenance activities should also be kept together with the current status of all maintenance teams.

2.1.10 Tracking and Genealogy of Products

In the manufacturing process of each order, MES should keep track of the equipments and materials being used in each product, operators and supervisors responsible for each step of the production, possible problems and the tools involved. After each step is finished, the information should be made available in a way that each sold product can be tracked to its components and processes.

2.1.11 Performance Analysis

The system should provide analysis reports in real-time of the indexes of performance analysis of the productive process, such as per-machine and per-operator productivity, quality parameters information, schedule information compared to what was planned, resources utilization and production cycle of each item. This data should be presented in *on-line* reports so that a continuous evaluation of the production can be made.

2.2 The Benefits of MES

One of the main characteristics of MES is that it is already used in a wide variety of companies, in a wide variety of fields, with positive results. Satisfaction about the system and the benefits achieved are obviously tied to the quality of the system employed but a research by MESA showed the following results for reference purposes:

- Reduction in the manufacturing cycle: 60% affirmed having obtained a reduction on the order of 40% or more;
- Reduction in data acquisition times: 63% of the companies stated having a reduction of 75% or more;

- Reduction in between-turns bureaucracy: 63% stated having a reduction in the volume of documentation exchanged between turns of 50% or more; and
- Reduction in delivery schedule: 50% stated having a reduction of 30% or more.

As already stated, MES' main character is to be the bridge between planning and shop floor, acting directly in the reduction of costs and production timing, rationalizing plant operation to the maximum. When a MES system is used to actuate on industrial processes in real-time, the data interface starts to feed the planning and management systems with up-to-date information, helping in cost reduction, labor payment, order requests, inventory and many other factors that are important to a company's success. (MESA, 1997a)

3. Microsoft-based distributed objects

3.1 COM/DCOM

The DCOM *(Distributed Component Object Model)* architecture was initially conceived as a component of the Windows DNA *(Distributed Internet Applications)* specification, that was the original attempt from Microsoft for creating a technical specification to allow interoperability in distributed systems. (FlashDee, 2002) Later, as Windows DNA evolved to .NET, where Microsoft proposes a diagram of distributed computing based on low coupling services (Microsoft, 2002), the DCOM model was kept as a part of the specification.

Figure (2) shows the COM standard is the integration infrastructure used to implement components that interact inside an address space, or in processes in a single host. COM is the basis over which are OLE *(Object Linking and Embedding),* ActiveX, MTS *(Microsoft Transaction Services)* and an ever growing number of services provided by Microsoft operating systems, like the multimedia DirectX are implemented. (Tallman et al, 1998)

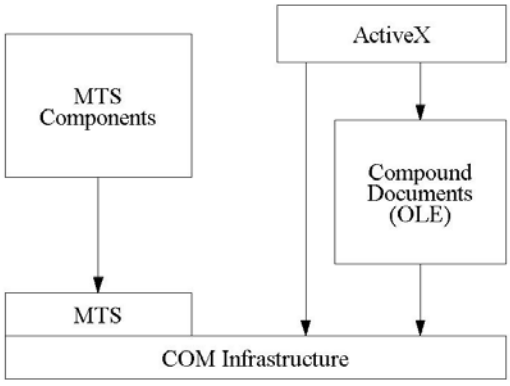

Figure 2: The COM Model. (Tallman et al, 1998)

DCOM is an extension to COM. DCOM builds a RPC *(Remote Procedure Call)* for objects (ORPC) over a pre-built RPC DCE *(Distributed Computing Environment),* intending to support the use of remote objects. This way, DCOM makes the interaction between objects executed in different hosts possible. The DCOM standard was introduced with the Windows NT 4.0 operating system, and as an additional to Windows 95 by the end of 1996.

A COM object can support multiple interfaces, each one representing a different vision of the behavior of the object and through these interfaces, client COM objects can relate to server COM objects. This is also what happens with CORBA. Still like CORBA, COM allows the integration of components written in different programming languages as C++, Java and Visual Basic. (Chung et al, 1998) DCOM operates in a client/server schema: to request a service, a client invokes a method implemented by a remote object that, this way, starts to act as a server. Differing from CORBA, DCOM does not support multiple heritage of interfaces, but allow the use of multiple interfaces by one same object, reaching the same result.

Comparing the CORBA and DCOM architectures, one can notice that they are very similar: both proportionate the infrastructure of distributed objects need for transparent activation and access to remote objects, but some differences exist: DCOM supports object with multiple interfaces, what does not exist in CORBA; CORBA supports multiple heritage by IDL, what DCOM doesn't; every CORBA interface inherits the constructor that realizes the implicit tasks like reference generation, in DCOM these tasks have to be done explicitly; DCOM has strong ties to RPC, CORBA doesn't; DCOM's specifications include many items that are considered details in CORBA.

3.2 OPC

Since 1998, the OPC Foundation, a non-profit organization, established a set of standard protocols for OLE/COM interfaces aimed at providing greater interoperability between automation/control applications, systems and field devices and office applications in the process control industry. This set of protocols makes the OPC *(OLE for Process Control),* based on the functional requirements of the OLE/COM from Microsoft. (OPC, 1998a)

OPC specification defines standard objects, methods and proprieties for information servers in real-time, like PLCs *(Programmable Logic Controllers),* intelligent field devices and sensors, to communicate the information from these servers to devices that are compatible with the OLE/COM technology, like supervisory software or report generators. In the conception of the OPC Foundation, there is a need for a common method for the applications to access data from any source, be it a device or a database. (OPC, 1998b)

There is a number of supervision and control client applications developed today that need to acquire data from external sources, like a sensor or a PLC and that leads to the need to develop specific drivers that make the interface between them. (OPC, 1998b) This driver development brings some problems: duplicated efforts, once that all software suppliers need to write a driver for one single hardware; inconsistency between drivers from different suppliers; lack of support for changes in the hardware; access conflicts and others.

The objective of OPC is to determine the frontier between hardware suppliers and software suppliers in a way that data from a source can be used by any other client application in a standard way. By this concept, a supplier can develop a reusable server component to communicate with its data sources (normally a piece of hardware) and makes an OPC interface available as a server so that any client can access data from this source.

OPC's first version focused in functionality commonly found in hardware suppliers for industrial control applications: real-time access to data, alarms and event manipulation and data history access.

OPC specifications include a set of custom COM interfaces for the use of clients and servers, as well as references to a set of OLE Automation Interfaces to support clients developed from higher-level applications, like MS-Excel or MS-Visual Basic. An OPC client can communicate with OPC servers from different suppliers, as the code supplied by the hardware or data source supplier determines which devices and data each server has access to and details about how the server can physically access the data. (See Fig. (3))

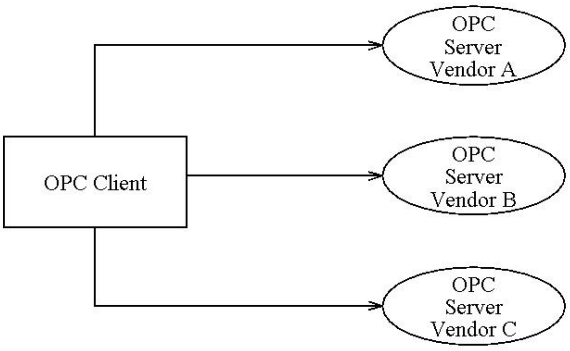

Figure 3: OPC Client (OPC, 1998b)

OPC interfaces can be used in many ways in an industrial control environment: since the direct extraction of data from physical devices for a SCADA *(Supervisory Control And Data Acquisition)* or DCS *(Distributed Control Systems)* system to the extraction of data from a SCADA or DCS for a higher-level application. The architecture allows an OPC server to be built that allows an application to access data from many OPC servers of different suppliers, running in different places in the network through one only object.

The OPC Foundation is working on a new generation of the OPC specification, based on Microsoft's .NET and XML *(Extensible Markup Language)*. However as this is work-in-progress, the specification is not available yet.

Another version of OPC, called *OPC Data Exchange* (OPC-DX) focus on the elimination of data bridges between the industrial *bus* protocols (like DeviceNet, Profibus, FieldBus and ControlNet), however, this is also work-in-progress as of this writing. The communication mechanism in OPC-DX is server-server. In it, devices like networks or subsystems connect to server components called OPC-DA *(OPC Data Acquisition)*, that in turn connect to each other directly, through OPC-DX interfaces. Communication is made in a peer-to-peer schema, eliminating the need for an application bridge (see Fig. (4)).

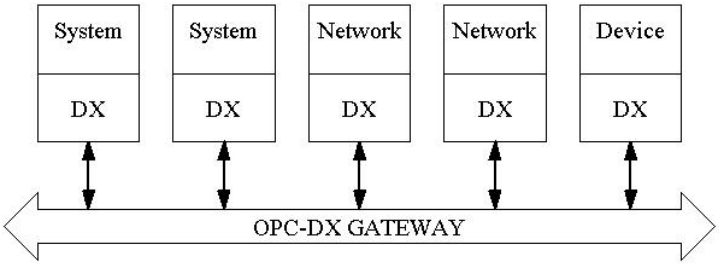

Figure 4: Communication of OPC-DX

4. CORBA

One of the basic concepts needed to have a good comprehension of the CORBA architecture, its capabilities and advantages reside in the Object Orientation paradigm. Object Orientation is an evolution from the structured programming methodology (implemented by some old programming languages like C and Pascal) and consists in a way of making a computer program as a set of objects that work together in a pre-defined way to make tasks. (Rumbaugh, 1994) Programs are composed by a set of components, called objects that actuate in a coordinated manner to answer to a given application.

Due to the advent of object-based applications and of distributed systems, and the resulting necessity of standardizing interfaces and of a series of essential services for those applications to effectively reach their objectives, the OMG *(Object Management Group)* was established in 1989.

The central objective of OMG consists in developing a common architecture for object-oriented applications based on interfaces specifications (Soley et al, 1995) through the establishment of OMA *(Object Management Architecture)*, of which CORBA is a component. Basically, OMA supplies a group of standards over which applications are developed, consisting of an ORB *(Object Request Broker)* function, object services (CORBAservices), common facilities (CORBAfacilities), domain interfaces and application objects. The role of CORBA inside OMA is to implement the ORB function, supplying a standardized mechanism for interface definition between components, besides tools to make the implementation of these interfaces easier making use of the programming language of user's choice. Two of the greatest proprieties of CORBA are platform independency and language independency.

The model of reference identifies and characterizes the components, interfaces and protocols that make OMA, including the Object Request Broker (ORB), the component that allows clients and objects to communicate in a distributed environment. Besides, the model of reference presents four categories of interfaces objects:

- Objects Services – interfaces for general services, with potential use by any application that is distributed-objects based;
- Common Facilities – interfaces for horizontal facilities (generic reusable applications), applicable to the majority of applications domains and focused in helping the user in the implementation;
- Domain Interfaces – application-domain specific interfaces; and
- Application Interfaces – non-standard, application-specific interfaces.

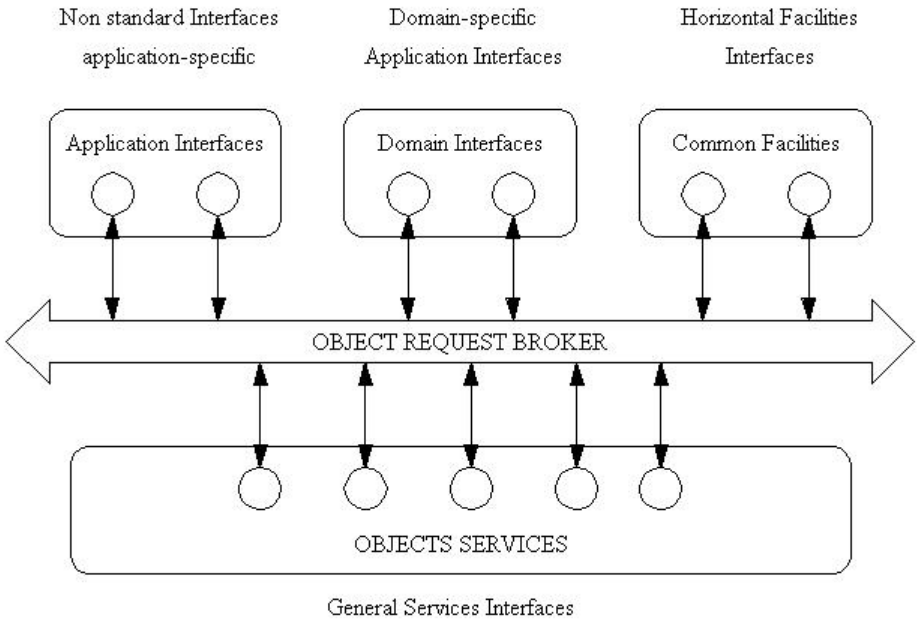

Figure 5: OMA reference model: interfaces categories.

4.1 The Object Request Broker

ORB, the fundamental component of CORBA, consists in a software element which purpose is to realize the communication between objects. This is achieved through making available a number of capabilities (OMG, 1998a), among them the capability of finding a remote object from a reference (see Fig. (5)).

An ORB is a mechanism used by objects to make requests and receive answers one from another, be it in the same host, be it from network. With ORB the client does not need to know of the mechanisms used to communicate to or

activate an object, its implementation properties or even its location. ORB is, therefore, the basis for building distributed applications and for making interoperability between applications in homogeneous or heterogeneous environments.

4.2 The Interface Definition Language (IDL)

Another fundamental component of the CORBA architecture is IDL. IDL's function is specifying interfaces between CORBA objects, being in great part responsible for the language independency proposed by the standard. IDL is the standard language used for defining interfaces between applications' components. However, IDL is not a procedural language: it can be used for defining interfaces, but not implementations. IDL can be compared to the header files used in C or C++.

IDL specification is responsible for assuring that data is correctly exchanged between different programming languages. (Rosemberg, 1998) For example, IDL has a *long* type that consists of a signed integer number with 32 bits that can be mapped into a *long* in C++ or into an *int* in Java. Once all interfaces defined by CORBA are described via IDL, they can be mapped into any implementation language: a server written in C++ can communicate with clients written in Java, that can communicate with servers written in COBOL and so on.

IDL, like the majority of other programming languages, defines primitive types that include integer numbers, characters, floating-point numbers, strings, Boolean types and constants, among others. Besides that, primitive types can be aggregated into new types, like *unions* and *structs*. Also defined by the IDL specification are parameters representations of input, output, methods, modules, attributes, heritage between classes and other tools needed for the complete description of the interfaces between various classes in a system.

4.3 Communication and CORBA Objects Models

To make communication between objects easier, CORBA uses the notion of object references, or IORs *(Interoperable Object References)*. When a component of an application wants to access a CORBA object, it initially obtains an IOR for that object, through which it will be capable of invoking methods from the objects. At that point, the object becomes a server. For CORBA, a client is simply an application that is using the services from a CORBA object, that is, invoking methods in another object.

ORBs normally communicate through the Internet Inter-ORB Protocol (IIOP). There are other protocols for inter-ORB communication, but IIOP is the most popular, (Orfali et al, 1998) due to the popularity of the TCP/IP protocol, a lower layer that is located right under the IIOP layer. However, CORBA is network-protocol independent, and could work with any other.

Visibility of CORBA objects is made only through reference passing to the objects: value passing is not possible (at least until version 2.0). So being, CORBA remote objects remain remote, it is not possible, for now, for objects to copy themselves to other places.

CORBA defines BOA *(Basic Object Adapter)* that makes the interfaces between objects and their ORBs, through the implementation of series of methods for accessing ORB's functions, like activation-authentication of an object or the persistence of an object.

4.4 Objects Services

To make the development work easier, OMG created the Object Services, which consist in fundamental services that supply a base for application development. Object Services are basic elements for application based on distributed objects, combining in different manners attending needs as object localization, persistency control, concurrency control, event treatment and others. Object Services already adopted by OMG are called in a general manner CORBAservices, and include the Names, Events, Life Cycle, Persistency, Transactions, Concurrency Control, Relationship, Externalization, Licensing, Proprieties, Safety and Time services. (OMG, 1998b and OMG, 2000)

4.4.1 Names Service

When a component of a given application wants to access a CORBA object, it first has to get an IOR for this object, though which it is capable of invoking its methods. The most basic manner of working with the IORs is generating small text files with pre-defined names and writing the IORs in these files. When an object needs to identify a second object, it can open the generated file and read its IOR, and then it will be capable of interacting with it. Nevertheless, that method needs that all the objects interested in writing IORs write in pre-defined places and that all the objects interested in accessing them should have access to.

To solve this object localization problem, OMG defined the Names Service. (OMG, 1998b) This service aims at associating names to the objects, using the *name bindings*. These associations are always defined in the so-called *names contexts*, which are objects that contain a group of names associations, each one being unique in a context. Different names can be associated to an object, in one same context or in different ones, but it is not mandatory that all the objects be assigned names.

4.4.2 Events Sevice

At each CORBA call between one object and another, parameters are generally passed and answers are generally returned: there is data communication between the client object and the server object. A standard call results in the execution of an operation by the server, realized in a synchronous way. The call is well succeeded if both, client and server are "online" and available; exceptions are raised in case this basic condition is not fulfilled. However, synchronous calls do not solve all the needs of a programmer or developer working in a distributed system. (OMG, 1998b) There are many situations when it is important to decouple the communication between client and server: for improving system performance, for making the system more reliable or for peculiarities to be solved by the system. Events Service was defined to promote this decoupling, (OMG, 1998b) allowing objects to communicate in asynchronous ways.

The Events Service defines two new roles for objects: the Supplier and the Consumer. Supplier objects are those that generate the events (produce the data) and Consumers are those that process these events or data that are passed to them by CORBA requisitions.

4.4.3 Notification Service

Events Service contributes substantially for making asynchronous communication faster between suppliers and consumers, but there are situations where this service shows deficiencies in its specification. Especially for applications with high reliability needs, as is the case of a system for shop floor supervision and control, there are some necessities that cannot be ignored: quality of service guaranties; a filtering system so that a client only receives the events in which it is interested; and the capability of transmitting structured data.

These necessities are supplied by the Notification Service, (OMG, 2000) that is an extension to the Events Service. The primary objective of the Notification Service is to make the Events Service better by introducing the concepts of filtering and configurability, according to needs from Quality of Service. Clients of the Notification Service can subscribe to specific events, associating filters to the proxies they use to communicate with the event channels. Filters encapsulate restrictions that determine the events in which the clients are interested in such a way that the channel only delivers events corresponding to the interests expressed by the clients.

5. Commercial MES

Table 1. Commercial MES products, manufacturing companies and available distributed objects.

	Company	Product	Website	OMG	OPC
1	ABB	Industrial	www.abb.com	X	X
2	Datasweep	Advantage	www.datasweep.com		
3	AspenTech	AspenTech Mfg. Suite	www.aspentech.com		X
4	GE Fanuc	Cimplicity	www.gefanuc.com	X	X
5	Ci Technologies	Citect Plant2Business	www.citect.com		X
6	Siemens ORSI	Cube Ind. Framework Suite	www.siemens-orsi.com	X	X
7	Brooks Automation	FactoryWorks	www.brooks-pri.com	+	
8	Teradyne	GR Force/SCE	www.teradyne.com	+	
9	Meteor-IT	Hacon Primas	www.meteor-it.nl		
10	HMS Software	HMS	www.hmssoftware.com		
11	Cimnet Inc.	Folders/Factelligence	www.cimnetic.com		
12	Camstar	InSite	www.camstar.com		
13	Intellution	Intellution Dynamics/iFIX	www.intellution.com		X
14	Mware	Mware MES	www.mware-mes.com		
15	Real World Tech	On Track	www.rwtcorp.com		
16	Optal Ltd.	OPTAL	www.optalmes.com		
17	OSIsoft	OSI-PI	www.osisoft.com		X
18	Interact Ind. Automation	PASSy	www.interact-automation.nl		
19	Werum Software	PAS-X	www.werum.de		
20	Rockwell Propack	PMX	www.propack-data.com	X	X
21	Honeywell	POMS	www.poms.com	X	X
22	Mountain Systems	Proficy for Manufacturing	www.mountainsystems.com		
23	IBASEt	Solumina	www.ibaset.com		
24	IBM	SuperPoseidon	www.ibm.com	X	
25	SynQuest	Virtual Production Engine	www.synquest.com		
26	Invensys	Wonderware InTrack	www.wonderware.com		X
27	USData	Xfactory/FactoryLink	www.usdata.com		X
28	Elan Software	XFP	www.elansoftware.com		

Table (1)'s objective is to identify the utilization of CORBA or COM/DCOM technologies in the listed products. However, as all of them make use in some way of COM/DCOM, the use of CORBA and OPC were compared.

The last two rows of Tab. (1) identify the companies that are members of the organization and offer at least one product based on the technology in question. Companies identified by a "+" sign are not members of the organization, but it was possible to identify at least one product based on CORBA or OPC.

All the companies listed classify their products as *Manufacturing Execution Systems*, and many also offer SCADA packages, usually as a subset of their MES packages.

While all the companies listed in Table (1) have products based on the COM/DCOM technologies, just a part of them offers solutions based on the OPC architecture, as is the case of Invensys´ Wonderware InTrack and USData´s Xfactory. This is due to OPC being more widely used in instrumentation and SCADA than in the MES packages.

6. References

Brown, L., 2000, "Integration Modeling Techniques for Enterprise Resource Planning (An ERP Case Study)", Sams Publishing.

Chung, P.E. et al, 1998, "DCOMxCORBA Side by Side, Step by Step, Layer by Layer", Control Engineering Website: www.controleng.com.

FlashDaddee, 2002, "The .NET Framework", 2002, Website: http://www.flashdaddee.com/Books-Technical/InsideCsharp/32ch02c.htm.

Highlander Communications, 1998, "CORBA in the embedded systems context", Highlander Communications, L.C.

Hoske, M.T., 1998, "Objects Make Software Behave like Hardware", Control Engineering.

Kamal, S.Z., 1998, "Integrated Enterprise proves key to flexible manufacturing", Intech, Vol. 45, Number 7.

Mello, A., 2002, "ERP Fundamentals", ZDNet, website: http://zdnet.search.com/search?cat=279&tag=st.ne.sr.srch.zdnet&q=%22IDC%22

MESA, 1997a, "The Benefits of MES: A Report From The Field", MESA International White Paper #1.

MESA, 1997b, "MES Explained: A High Level Vision", MESA International White Paper #6.

MESA, 2001, "Definition of MES", MESA Website: http://www.mesa.org/html/overview.html.

Microsoft, 2002, "What's different? Innovating a Third Generation of Computing", Microsoft website: http://www.microsoft.com/net/defined/netchange.asp

OMG Manufacturing Special Interest Group, 1996, "Manufacturing Enterprise Systems, A Whitepaper", OMG Documet Number mgf/96-01-02.

OMG Manufacturing Domain Task Force, 1997, "RFI-3 Manufacturing Execution Systems (MES)", OMG Document Number mfg-97-11-01.

OMG, 1998a, "The Common Object Request Broker: Architecture and Specification", OMG Document Number 98-07-01, Revision 2.2.

OMG, 1998b, "CORBAservices: Common Object Services Specification", OMG Document 98-12-09.

OMG, 2000, "Notification Service Specification", OMG Document 00-06-20.

OPC Task Force, 1998a, "OPC Common Definitions and Interfaces", version 1.0, OPC Foundation Document.

OPC Task Force, 1998b, "OPC Overview", version 1.0, OPC Foundation Document.

Orfali, R. et al, 1998, "Client/Server Programming with Java and CORBA", 2nd edition, John Wiley & Sons.

Rosemberg, J., 1998, "Teach Yourself CORBA in 14 days", 1st edition, Sams Publishing.

Roy, C. and Johnson, L., 2002, "Positive Yield", InTech, Vol. 49, Number 4.

Rumbaugh, J., 1994, "Modelagem e Projetos Baseados em Objetos", Editora Campus, Rio de Janeiro.

Soley, R.M. et al, 1995, "Object Management Architecture Guide", 3rd edition, John Wiley & Sons.

Tallman, O. and Kain, J.B., 1998, "COM versus CORBA: A Decision Framework", Distributed Computing.

7. Copyright Notice