26

Integrating Infrastructures for Manufacturing a Comparative Analysis

José P. O. Santos[1], João J. Ferreira[2], José M. Mendonça[3]
1University of Aveiro / Dept. of Mechanics / 3800 Aveiro, Portugal
Tel.: +351 34 370892, Fax: +351 34 370953, email: jps@inesca.pt
2,3FEUP-DEEC/INESC/ Rua José Falcão 110, 4000 Porto, Portugal
Tel.: +351 2 2094300, Fax: +351 2 2008487, email: jjpf@inescn.pt

Abstract

This paper presents an overview of the evolution of manufacturing systems integrating infrastructures and standards, ranging from industrial network standards[1] such as MAP[2] and CNMA[3], that essentially support hardware integration, distributed platforms like OSF/DCE[4] and OMG/CORBA[5] promoting industrial applications integration, up to integration frameworks like CIMOSA[6] which try to encompass all the aspects linked to business integration. This overview has a particular emphasis on the integrating infrastructure concepts towards enterprise integration.

The advantages and constraints of a CIMOSA integrating infrastructure implementation using OSF/DCE is also presented and discussed.

Keywords

Integrating Infrastructures, CIMOSA, DCE, CORBA, MAP, MMS

1 INTRODUCTION

Enterprise integration is becoming a must in competitive manufacturing, particularly when improving delivery performance, time-to-market reactivity and flexibility is the issue.
The enterprise's activity integration can be made at three levels:

- Physical systems integration: providing physical interconnection and reliable communication between enterprise resources, through the use of *industrial networks* standards, like MAP an CNMA. Despite industrial network's standardisation efforts, it is hard to promote physical systems integration at all levels.
- Applications integration:making use of distributed software for industrial automation overcoming: hardware, operating systems, networks, programming languages and data base heterogeneity. It is necessary to use a standard and portable packet of software (*middleware*)

[1] Open System Interconnection (OSI) based.
[2] Manufacturing Automation Protocol (MAP).
[3] Communication Networks for Manufacturing Application (CNMA), It is not yet a standard.
[4] Open Software Foundation/Distributed Computing Environment (OSF/DCE).
[5] Object Management Group/Common Object Request Broker Architecture (OMG/CORBA).
[6] Open System Architecture for CIM (CIMOSA).

L. M. Camarinha-Matos (ed.), *Re-engineering for Sustainable Industrial Production*

that provides a common application program interface (API). This middleware must provide not only communication services but also a set of common distributed services, to be used in the development and operation of all distributed applications (planning, CAD, CAM, MRP, shop floor applications, sales, etc.).

- Business integration: The business integration necessity comes from the fast market evolution imposing a continuos enterprise adaptation to rapid changing product specifications and production strategies. Despite the technological advantages offered, by present distributed platforms advantages in hardware and applications integration it is important to have fast means to analyse, design and implement new functions and operations, as well as to have production process modelling and simulation [8] integrated with production control and monitoring, as defined by *CIMOSA* architecture.

Within the context above this paper presents and compares the advantages and constraints of, industrial networks (MAP, EPA), distributed platform (DCE), CIMOSA architecture, and attempts to answer to the following two questions:

- Which relationship, advantages, and constraints have distributed platforms and industrial networks?
- Keeping in mind that we want to find a general Information Technology (IT) platform able to support CIMOSA architecture, w*hich advantages and constraints have industrial networks and distributed platforms to do it ?*

2 INDUSTRIAL NETWORKS [1][2]

Keeping in mind the integration of activities we present some communications protocols [1,2,3], its relationships with the OSI reference model [4], and its advantages and limitations.

Since the 60's computers began to be used in industry for Accounting and others Management functions, Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Computer Aided Engineering (CAE), Computer Aided Production Planning (CAPP), Supervisory Control And Data Acquisition (SCADA), etc.

The exponential growth of the amounts of information, that had to be exchanged and stored, together with organisational changes imposed by the rapid market evolution (small series of many products), led to the unfeasibility of traditional methods (like diskette, tape, paper and informal communication).

In the 80's better integration of these enterprise functions was attempted (fig. 1) by several suppliers, by using proprietary networks and protocols (fig.1).

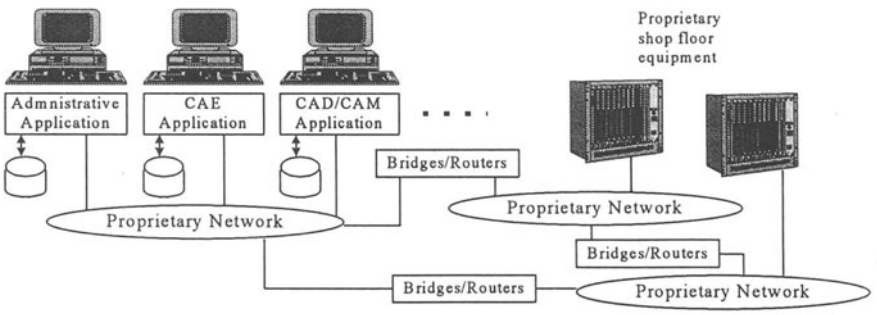


Figure 1 Networking: The first level of integration.

Despite the good results achieved to reach the actual integration of activities in an environment with multiple pieces of proprietary equipment and networks has been a difficult and expensive task. The only way to have a maximum level of integration is the adoption of communication (networks and protocols) standards, like MAP [5] (fig. 2) and CNMA [6], cost issue however still unresolved .

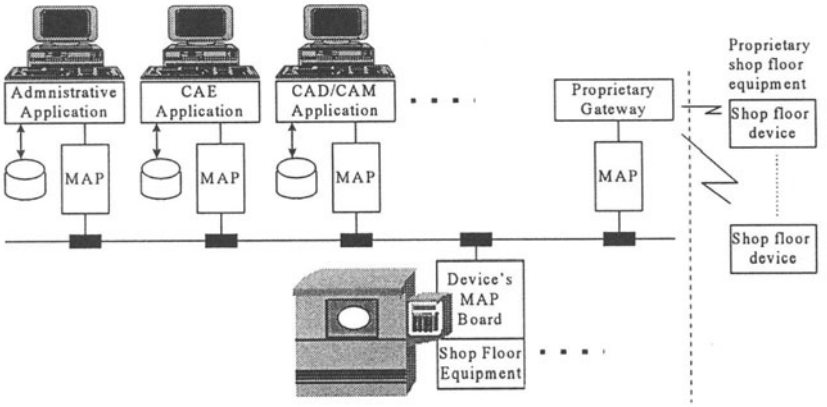
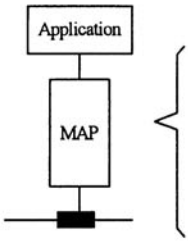


Figure 2 Using MAP interfaces in manufacturing integration.

Both MAP and CNMA standards selected a sub set of pre-existing communication protocols according his specifics needs to some or all, OSI layers. For example MAP (fig. 3) uses the following protocols.



OSI Layer	MAP ver. 3.0
7- SASEs CASEs	ISO-8571(FTAM), ISO-9506/1/2(MMS),...
6- Presentation	ISO-8822(3)/ASN.1 ISO-8824(5)
5- Session	ISO-8327
4- Transport	ISO-8073 (class 4)
3- Network	ISO-8473 (Internet Protocol, connection less)
2- LLC	ISO-8802/3 (type1, class1) / IEEE 802.2
MAC	ISO-8802/4 (Token bus) / IEEE 802.4
1- Physical	ISO-8802/4 (Bus, Broadband Cable-CATV)

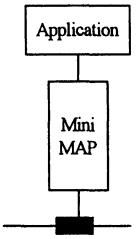
Figure 3 MAP protocols.

As figure 3 portrays MAP uses many communication protocols which require some processing capacity on every network node. However, many of the manufacturing components have a limited processing capacity, and therefor on the third MAP revision a new communication standard (mini-MAP) called Enhanced Protocol Architecture (EPA) was defined. EPA architecture (fig. 4) is much more easily implemented than MAP because it has only three layers: application, link and physical layer.

Despite MAP and EPA resemblance, a bridge it is necessary to interconnect both industrial networks because they use different modulation signals on physical level and consequently different medium access control in the link layer.

The third level of EPA (application level) interfaces directly with the manufacturing applications.

Thus EPA communication protocols do not provide network and transport services thus lowering message transmission reliability only supports the transmission of a limited number of bits per message. EPA does not provide session services (ex. dialogue synchronism points) neither presentation services (ex. ASN.1).



OSI Layer	MiniMAP ver. 3.0
7- SASEs CASEs	ISO-8571(FTAM), ISO-9506/1/2(MMS),...
6, 5, 4, 3	Null
2- LLC MAC	ISO-8802/3 (type1, class1) / IEEE 802.2 ISO-8802/4 (Token bus) / IEEE 802.4
1- Physical	ISO-8802/4 (Bus, Carrierband)

Figure 4 MiniMAP protocols.

From the figure 4, one can say that EPA defines an application layer directly on the top of a Local Area Network.

In order to allow the connecting a device into the network, a computerised controller with a network interface board must be used.

Usually the services provided by both the physical layer and the medium access control sub layer are implemented by the network board firmware. The others layers of MAP, EPA, CNMA, etc. are implemented by software routines which are ,in turn, used by the application software. These routines are executed, in the device motherboard, and they usually access the interface network board services using its I/O address, memory and interrupts.

One can say that the greatest contribution of MAP or EPA to manufacturing activities integration was the definition of MMS (ISO9506/1/2).

However, there are others ways: to transmit information through the factory with an higher security, to synchronise the enterprise activities, to co-ordinate factory resources and others ways to locate orders and products.

3 DISTRIBUTED PLATFORMS

Industrial network standards were just a first step within the scope of the global integrating infrastructure. Distributed platforms are another important step to promote a really distributed manufacturing integration infrastructure.

Nowadays, because of the enormous quantity and diversity of information, as well as hardware and software components generating processing and using it, not only at the shop floor level but also at all enterprise levels an encompassing enterprise integration framework is necessary beyond industrial network's standardisation.

In order to develop distributed systems for industrial automation overcoming hardware, operating systems, networks, programming languages and data base heterogeneity, is necessary to use standard and portable middleware providing a common application programming interface (API) that will easy the, development of applications (planning, CAD, CAM, MRP, shop floor control, sales,...). These manufacturing management applications must co-operate, share information and resources. A middleware structuring providing a set of common and distributed services to be used by all these distributed applications is therefor required.

These distributed platforms must overcome several problems:

- A CIM system is composed of several applications that must communicate and store information in a standard format.
- CIM applications must deal with high number of PCs, minicomputers, mainframes and shop floor devices produced by many different manufacturers.
- Global and distributed information access with appropriate security levels should be provided.
- Machines, computers and applications, may crash. On a distributed platform, if one equipment breaks, another one must replace it while providing exactly the same services (reliability and fault - tolerance).
- Development and maintenance costs of distributed manufacturing applications tend to be high.
- The hardware components of an enterprise must be located wherever they are needed (flexibility).
- An enterprise must be easily upgraded with more and new components to satisfy ever changing requirements (scalability, upgradability).

As to overcome the problems above referred, distributed platforms must provide a set of useful common distributed services, such as:

- Standard communication and distributed service protocols :
- Standard data representation:
- Directory service: a distributed system will have a high number of enterprise resources and the data associated with them to be stored in several places that can change dynamically from one place to another. A directory service can help providing the resources locations and properties.
- Security service providing several services to verify and control the access to shared resources.
- Distributed time service: Each enterprise has many resources with their own clock to synchronise their individual activities. To allow for an integration of enterprise activities one must have means to synchronise them. This is a problem, for example, for distributed applications that care about the sequence of events if each resource has a different notion of the current time.

Nowadays, several computing environments already use distributed platforms as middleware between the applications and specific hardware/networks. *One possibility is to use these platforms also in industrial environments*, alone or together with industrial network protocols (MAP,EPA,..), and in this later case *to choose which is the best distributed platform to reach integration.*

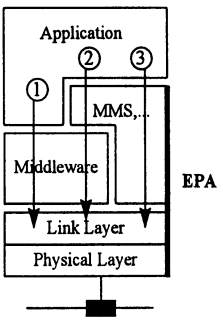


Figure 5 Middle ware and EPA integration.

As illustrated in the fig. 5 such middleware can be used together with existing industrial communication protocols. As the middleware needs some computational capacity we recommend to use, if possible, EPA industrial communication protocol. As figure 5 shows, an application can use the middle ware services alone (1), the EPA application layer services alone (3), or even MMS messages inside the middleware messages (2). In turn, both middleware and EPA application layer will use the EPA' Local Area Network (Physical layer and MAC sub layer) to send data through the physical transmission cable.

The middle ware provides a set of software tools and defined services, that make it much more easy to develop and operate distributed application. The middleware runs on the top of the operating system and networking software of the computerised equipment therefore hiding resources heterogeneity. A great number of funcionalities are provided, that are used by applications and other software, that in turn are used by human operators.

Basically there are two generic middle ware standards, which are not proprietary, available in the market with a set of services, commercial support, and distributed platform products. These are the OSF/DCE and the OMG/CORBA.

These platforms use a client server architecture. When a device makes a request, a server or several servers return responses. To each request made by a software module running on the client side there is an associated software module running on the server side, that processes and returns the right results to the client.

3.1 OSF/DCE

DCE [7,8,9] supports both portability and interoperability by providing the developer with capabilities that hide differences among the hardware, software and networking elements.

As illustrated in fig. 6 DCE has seven components (services), that are described bellow:

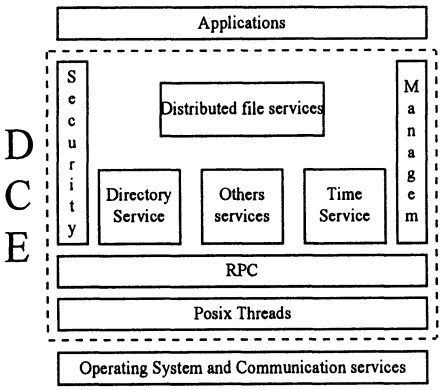


Figure 6 DCE components.

- The DCE Remote Procedure Call (RPC) provides a mechanism for communication between software modules running on different systems (figure 7), much simpler to code than older methods, like socket calls. RPC is probably the most important component of DCE, because all communication between clients and servers are performed using the DCE RPC protocol. On each call there is one or several parameters (data) that are transmitted to the server. The data transmission format used by DCE RPC is called Network Data Representation (NDR). The call parameter (data) are is first of all marshalled using client stubs (software routines). The client

stub gathers together all of the arguments (in or out arguments), pack them into a message, and sends it to the server, via network services provided by client runtime software. The server that has the corresponding server side stub receiving the invocation message, pulls out the arguments (unmarshalling), and calls the server procedure. When this procedure returns, the returned parameters (data) are marshalled by the server side stub into another message, which is transmitted back to the client side stub, which pulls out the data and returns it to the original caller.

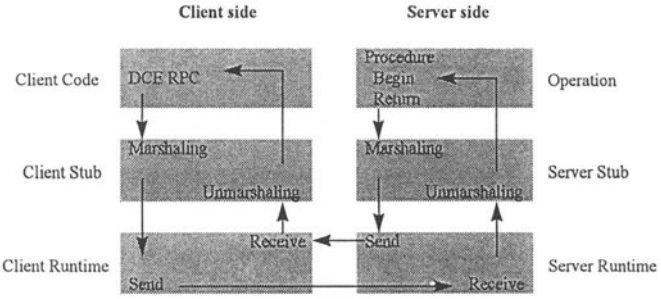


Figure 7 RPC execution steps.

- The DCE Directory Service, has three main components: Cell Directory Service (CDS), Global Directory Service (GDS), and Global Directory Agent (GDA).
- The DCE Security Service, is based on Kerberos (version V) and POSIX 1003.6 ACLs, it provides: user identification and authentication, access authorisation, secure RPC with several authentication options, data integrity and confidentiality.
- The DCE Distributed Time Service, provides services to periodically synchronise the clocks of different enterprise resources, and synchronises them with an external and correct time.
- The DCE Threads: Threads and Processes have a similar behaviour, although they are different. Each process has its own memory and CPU time. Each thread has its own CPU time, but *they share the same memory and resources.* It is possible to have several threads inside each UNIX process sharing the same memory and resources, but not CPU time. DCE middle ware provides the means to have one or several threads for each application on a single operating system task like MS-DOS or inside one UNIX process. DCE threads are based on the API interface for multithreaded programming draft 4 of POSIX 1003.4a.
- The DCE Distributed File Service provides a way to share remote files anywhere in the network. Using DFS remote files appear like local files to the applications.
- The DCE Management Service, can be used by all other services and applications providing services such as: event notification, software installation, distribution and maintenance services, and software licensing control.

Development process

There are several steps to develop a DCE application. First of all the application programmer, using the DCE Interface Definition Language (DCE-IDL), defines a RPC interface, and its associated data types. An interface is a group of operations (RPCs) that a server can perform. The definition of a RPC is similar to a local procedure with hers input and output parameters, except that in DCE-IDL only the calling interface is defined, not the implementation of the procedure side.

The DCE-IDL file is created with a text editor using IDL, a language that uses a C-like syntax to declare the attributes of the interface, its operations, and parameters.

Next, the programmer compiles the DCE-IDL file using the IDL compiler. The compiler produces output either in a conventional programming language, C language, or object code. The output of the compilation consists of a client stub, a server stub and a header file.

The #include (.h) file contains functions prototypes for all the operations and other data declarations defined in the interface, and must be used by both the client and the server.

The programmer then writes the client application code using the RPCs defined in the DCE-IDL file. Thus the client stub code must be linked with this application code, to perform a task behaving like a local procedure call, but that will in fact be performed on the server side of the application.

For the server side of the application, the programmer writes application routines that implement he operations defined in the DCE-IDL file. Thus, the server stub generated by the DCE-IDL compiler must be linked with the server application code. The server stub unpacks the RPC parameters and makes the call to the application as if the client program had called it directly.

3.2 OMG/CORBA

The Object Management Group (OMG) is defining a standard architecture for interoperable distributed software equipment, called Common Object Request Broker Architecture (CORBA). This architecture provides a high level of abstraction and multiple object oriented services to support distributed applications. Within this distributed object oriented platform client server architectures may be built, by allowing clients to issue requests through a local object interface accessing services supplied by the remote object implementation.

They are quite similar to C++ objects where each object's method definition is made on one header file and its implementation on other file. In this client server architecture (fig. 8) the object interface (method definition) is defined and used in the client side of one application. The object implementation (server side) providing the intended service, is used by the application server side, returning the results to the client side. This object interfaces are defined using the so-called OMG Interface Definition Language (IDL), that is a general object oriented language used to create application program interfaces. The OMG/IDL files look likes C++ header files and can be converted on specific languages like as C, C++, e Smaltalk.

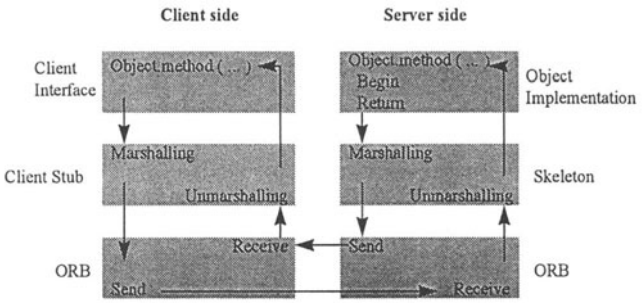


Figure 8 Object Interface / Object Implementation.

This architecture also defines how to locate (the right server) and to access (via ORB) the object implementation, but first of all the application client side must to know the object interface names.

- Interface Repository - IR: This repository keeps the objects interfaces names. The object's name is the only way for an application locate and access its services. This means that each object must write its interface's names on a IR.
- Object Request Broker - ORB: As presented, on a distributed client / server architecture, an application must have information about the interface name of the object's method (service) and a communication mechanism to access it. To this end, the OMG defined the Object Request Broker, which provides a set of high level communication services, being a CORBA main component. It is responsible for the transmission and routing of the client request through the distributed environment to a particular server, and for returning the results to the client. Through the ORB an application can remotely execute procedures as such as execute local procedures. The stub, the skeleton and the header files are generated by IDL specification compiler. The parameters of each object method are first of all marshalled using the client stub code. Each client stub is a piece of code that gathers together all of the arguments (in and out arguments), of its interface, and packages them into an adequate format message that it sends via ORB, to the server.
- Skeleton : On the server side there is a peace of code known by Skeleton Is the Skeleton, when the Skeleton receives a request, via ORB, from the Client, it calls the right object implementation. This object can already exist on a library or can be created by one user.
- Dynamic Interface Invocation - DII: There are two ways to call an interface (an object method): a static way, where the number and type of parameters pre defined at compile time, and a dynamic way where they can change dynamically at run time, namely the interface name it self.

Both DCE and CORBA architecture intended to be a middleware interface providing more or less similar services such as client/server communication, services to handle security and authentication, services to locate servers, and so on. A detailed comparison analysis between DCE and the CORBA, as well an evaluation of a CIMOSA integrating infrastructure implementation using the CORBA architecture are beyond of the scope of this paper.

4 CIMOSA REFERENCE ARCHITECTURE [5][9]

Keeping in mind that we wish to present the advantages and constraints of a CIMOSA integrating infrastructure implementation using either distributed platforms (DCE) or industrial networks (MAP) middleware. We will present a CIMOSA overview [10,11] with particular emphasis on the CIMOSA integration infrastructure requirements.

Despite distributed platforms advantages (hardware and applications integration) it is important to have fast means to analyse, design and implement new operations/processes. It's also important to have production and process simulation [8] integrated with production control and monitoring.

An ideal solution would be to be able to generate automatically from new market requirements for a new product all necessary programs to design and produce it.

An increasingly feasible approach consists of building on an enterprise computer representation (computer models) that would permit at first to simulate all daily enterprise activities. With such environment, simulation of the controlling components of a manufacturing system would be then possible. The new application programs would be first of all tested on the simulation environment and afterwards, using the same integrated environment, loaded on shop floor equipment to be used on the daily production.

To reach this goals CIMOSA provides, not only the integrating infrastructure concept, but also other two inter-related concepts: the Life Cycle (which deals with the development and

maintenance of enterprise and product models) and the Modelling Framework (enterprise models development process and reference architectures). CIMOSA is an architecture that aims to integrate, not only industrial hardware and applications but also enterprise activities (business integration).

The need for business integration comes from the fast market evolution which imposes a continuous enterprise adaptation to quick changes in product as well as in process specification, evaluation and implementation.

4.1 CIMOSA System Life Cycle

CIMOSA sub divides this integrated environment into two parts: development and maintenance of enterprise models (*enterprise engineering environment*), and the day to day control and monitoring of the enterprise's activities (*enterprise operation environment*). CIMOSA *enterprise operation environment* provides support to execute and control the enterprise activity through the *particular enterprise operation implementation description model*. Both environments use the CIMOSA *integrating infrastructure*.

A clear separation between the enterprise engineering and the enterprise operations concepts help us to simulate and evaluate new enterprise models without affect the day-to-day production.

4.2 CIMOSA Modelling Framework

To define the enterprise models (*enterprise engineering environment*) many CIMOSA development steps are necessary. The model creation processes will now be presented in reverse order. We begin from the particular enterprise operation implementation description model, used to control and monitor the day to day operations. The development of particular enterprise models is then discussed, up to the reference architecture generic building blocks used to define the enterprise models.

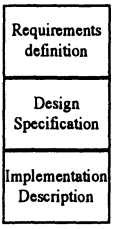


Figure 9
Particular enterprise model Levels.

The particular enterprise operation implementation model to be used in the enterprise operation environment is released from the enterprise engineering environment, more exactly from particular enterprise models.

Particular enterprise models (fig. 10) development is sub divided into three inter-related model levels which correspond to different life cycle phases.

At the so called third level (**analysis of requirements**) based on the market and enterprise goal's, functional specification is built using a user friendly language (particular requirement

definition model). This model defines the main events, data and macro operations. However at this level the model does not yet consider issues like the implementation environment or technology constraints.

At the second level (**design**), one allocates the previously defined functional specifications to each ,implementation independent enterprise resource. Using a computer processable language one defines how to synchronise and allocate the several enterprise operations to each resource, how to recover from some resources fails, etc. (*particular design specification model*). This model can already be use for simulation purposes. However at this level the model does not consider yet aspects like specific characteristics of proprietary resources.

At the first level (**implementation**), one defines commercial enterprise hardware and information technology specific assets as well several specific implementation details to enterprise operation (*particular implementation description model*).

During the development of each of these three levels is important to consider several enterprise aspects (fig. 11) necessary to reach enterprise requirements, like:

Function view: which planning, control and monitoring function and system behaviour are required. To develop functional and behavioural enterprise models CIMOSA proposes techniques such as SADT, IDEF0.

Resources view: which equipment, application, databases, human and communication functional entities are needed.

Information view: which data and data bases are required. To develop information models, CIMOSA proposes semantic object-oriented modelling techniques, entity-relationship diagrams, and logical and physical data models. The global information modelling framework is compliant with the three-schema approach proposed by ANSI/X3/SPARC, 1976).

Organisation view: who is the responsible for each: function, resource, information and exception, definition and maintenance.

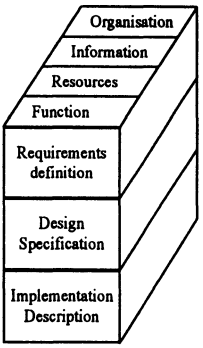


Figure 10 Particular enterprise model (Levels and View).

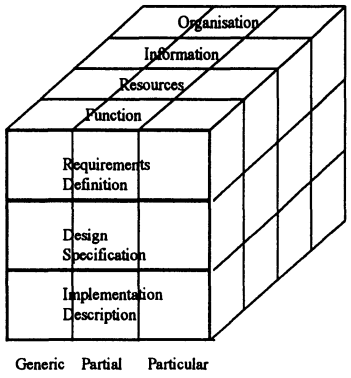


Figure 11 CIMOSA Modelling Framework (Levels, View, Generecity).

Until now we only (almost) speak about **particular enterprise models**, however CIMOSA provides (fig. 11) a partial and a generic set of building block (*reference architecture*) that we must use to get particular models.

CIMOSA **generic** building blocks :The generic building blocks are the basic blocks of CIMOSA architecture. These blocks are called "generic" in the sense that can be reused to build the models of all types of enterprises.

CIMOSA **partial** building blocks uses a sub set of generic blocks organised in set of incomplete models, each set to one type of enterprise. Each set of models can be then refined to get one set of particular enterprise models.

4.3 CIMOSA Integrating Infrastructure [12,13]

In order to allow enterprise implementation description models (and its functional operation - FO) simulation and execution *we need an integrating infrastructure that provides not only, a homogeneous and distributed access to all enterprise resources already achieved by industrial networks (communication infrastructure, abstract manufacturing services) and distributed platforms (global information infrastructure, resources location, data replication , access security, resources synchronisation)*, but also additional services required to execute enterprise CIMOSA models.

CIMOSA integrating infrastructure provides five entities (fig. 12) and each of them provides several inter related services:

- Business entity: is composed by the Business process control, resources, management and activity control part. They provide services to monitor and control enterprise operations. These services are used, for example, to execute and control the released Particular Implementation Description Model.
- Presentation entity:provides machine, functional and human dialogue services. These services are used by both distributed applications and the other entities.
- Information entity: provides data access, data integration and data manipulation services. These services are used by business and presentation entities. It's composed by system wide data services and data management services.
- System management entity: provides all network and system management services. These services are used by the other entities to perform tasks such as : to install and distribute new release of components, fault and performance management.
- Common services entity: It provides a set of common communication (exchange message, priority, transparency,...) and distributed services (naming, security, time,...). These services are used by the other entities.

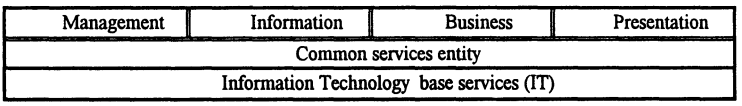


Figure 12 Integrating Infrastructure - five entities.

However the better suited CIMOSA information technology base services (IT), must be able to satisfy as much as possible the CIMOSA IIS requirements[14]:

- Distribution: The IT base services ought to provide distributed processing and distributed data access to the manufacturing and information systems enterprise's resources, connecting them via communications networks. The IIS itself ought be distributed.
- Openness: The IT base services are built of many components that must be integrated, working together. To achieve an higher degree of integration each component must (be Open) to provide standard services through a well defined and documented interface.
- Portability and connectivity support: An IT base service should allow a software enterprise model to be easily ported from one device to another without any problem (portability) and easily connected to any machine (connectivity).
- Portability of an integrating infrastructure: The IT base services themselves must be portable to any device, regardless of the devices on which they are presently running: operating system, language, etc.
- Use of existing standards: It is desirable that an IT base service is defined in compliance to standards, as to allow for portability and openness.
- Reliability: The IT base services should provide the means to recovery from faults or failures in the distributed environment components, such as retrying operations, duplication of used components, etc.
- Performance: The IT base services should provide an high performance.
- Migration support: The IT base services itself and the upper distributed system must support a fast time-migration.
- Conformance/compliance with the framework for integrating infrastructure: The IT base services ought to overcome the business, presentation, information, management and common services entities needs referred to above.

5 CONCLUSION

A few "questionable issues " and their answers can be aligned under the form of concluding remarks.

Which relationships, advantages and constraints, have distributed platforms and industrial networks ?

As presented, one can use both distributed platforms and industrial networks, each of them separately or together, to reach a greater integration of enterprise's activities (both engineering and operation environments). Thus, the question is therefore which is the better suited to each environment.

Both distributed platforms and industrial networks provide network communication facilities. As presented in figure 5, one can separately use (1,3) both middleware on the same device or even sends MAP-MMS messages as DCE-RPC parameters (2). However, MAP industrial network is better suited to be used in the shop floor environment than distributed platforms because:

- Most of distributed platforms capabilities, such as dynamic resources location, shared resources, dynamic task allocation and others, will not be useful to the shop floor activities because shop floor resources, physical layout, as well as material flow specificities do not allow it.
- Only few and expensive shop floor resources are able to support (i.e. run) distributed platform middleware.
- The MMS messages and the file transfer protocol provided by MAP are the most useful facilities to the shop floor activity integration.

Distributed platforms are, on the other hand, obviously better suited to improve the performance of Computer Aided Engineering Environment (CAEE) activities than industrial networks.

So within the context above, one can say that DCE and MAP middleware are complementary.

Table 1 CIMOSA IIS requirements

CIMOSA IIS requirements	MAP	DCE
Distribution	poor	very good
Openness	good	good
Models portability	poor	good
IIS portability	good	very good
Standards	*de jure*	*de facto*
Reliability	poor	very good
Performance	good	heavy
Time-migration support	poor	good
IIS compliance		
-business	--	--
-presentation	ASN.1	--
-information	poor	poor
-management	--	combined
-common	poor	OSF/DME
services		very good

Keeping in mind our objective of finding a general IT platform able to support CIMOSA architecture, which advantages and constraints have industrial networks and distributed platforms to do it ?

MAP, DCE, and CIMOSA IIS requirements have been presented, thus we are in position to compare them and conclude, which advantages has DCE over MAP to support CIMOSA-IIS. To

implement the CIMOSA IIS one could use distributed platforms or MAP. However, as presented on the table 1, MAP will not provide most of CIMOSA IIS requirements which implies that we would have to implement them on the top of MAP. On other hand, as DCE is much more complete it frees the CIMOSA IIS from having to implement most of IT basic services.

Some DCE constraints are nevertheless recognised:

- DCE is only a standard de facto not a standard de jure.
- DCE only provides a Network Data Representation (NDR), which is not enough to support the required presentation entity services.
- DCE itself does not provide management services; however OSF also defined another complementary standard (de facto), the Distributed Management Environment (DME), which combined with the DCE would provide most of the CIMOSA management entity requirements.

One can conclude that DCE provides most of the Common entity services and most of the general IIS requirements. However it does not provide any CIMOSA business or presentation services.

6 REFERENCES

[1] G. Messina, G. Tricomi, "Software standardisation integrating industrial automation systems",Computers in industry 25/(1994) 113-124.

[2] S. A. Koubias, G. D. Papadopoulos, "Modern Field bus communication architectures for real-time industrial applications", Computers in industry 26/(1995) 243-252.

[3] Fred Halsall, "*Data Communications, Computer Networks and OSI*", Addison-Wesley, 1988.

[4] Etienne Lapalus, Shu Guei Fang, Christian Rang, Ronald J. Van Gerwen, "Manufacturing Integration", Computers in industry 27 (1995) 155-165

[5] Zoubir Mammeri, Jean-Pierre Thomesse, "Résaux locaux industriels", Eyrolles, 1994.

[6] Esprit Project 7096, "CIME Computing Environment Integrating a communications network for manufacturing applications", (CCE-CNMA), 1994.

[7] OSF, "Introduction to OSF DCE", Prentice-Hall, 1994.

[8] Thomas W. Doeppner Jr., "A Technical Introduction to DCE for Managers", OSF/DCE Conference, London, UK, April 25-27, 1995.

[9] Harold W. Lockhart, "OSF DCE Guide to Developing Distributed Applications", McGraw-Hill, 1994.

[10] Esprit consortium AMICE, "CIMOSA Open System Architecture for CIM".

[11] Kurt Kosanke, "CIMOSA-Overview and status", Computers in Industry 27(1995)101-109.

[12] IFIP transactions Towards world class manufacturing, "Architectures for integrating manufacturing activities and enterprises", T.J.Williams, et al, 1993.

[13] IFIP transactions Towards world class manufacturing, "CIMOSA: Integrating the production", Jackob Vlietstra, 1993.

[14] CEN Report/TC 310, "CIM Systems Architecture- Enterprise model execution and integration services (EMEIS)"-"Annex E: CIMOSA 's Framework for integrating Infrastructure", CR 1832:1995, Feb.1995.