Communicating to a Manufacturing Device using MMS/CORBA

T.Ariza, F.J. Fernández and F.R.Rubio
matere@trajano.us.es fjfj@trajano.us.es rubio@cartuja.us.es
Dept. Ingeniería de Sistemas y Automática.
Escuela Superior de Ingenieros. Univ. de Sevilla. Spain

Abstract: The advantages of distributed systems are also applied to manufacturing systems due to their inherent distribution. However, the heterogeneity found in the current hardware and software and the underlying communication between the different components of the system make the development of these systems a difficult task. MMS allows a uniform communication with different hardware systems. On the other hand, CORBA makes the communication between several objects easier, reducing the implementation cost to a minimum. By joining both technologies a communication method between devices can be achieved. This method is location transparent and specific physical features independent. The MMS implementation over CORBA allows new devices to be added in a natural way, making use of the inheritance available in object oriented programming and the facility supplied by CORBA in the communication. In this work, this implementation has been used in order to communicate to a real device.

Keywords: Distributed Manufacturing System, Object Oriented Programming, CORBA, MMS, JAVA.

1. INTRODUCTION

Distributed systems have been widely introduced in manufacturing due to their flexibility, reliability, incremental growth and better price/performance rate. But the main problem that arises is the communication between several interconnected elements, which are very different.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35492-7_50
G. L. Kovács et al. (eds.), *Digital Enterprise Challenges*

MMS is an application layer protocol that homogenises the use of the devices that compound the system. MMS makes use of this approach to specify the services that a user can invoke to communicate with these devices. On the other hand, distributed object methodologies, where CORBA is framed, provide all the object oriented methodology advantages in distributed systems. CORBA can make the MMS service user application to request the service easier, as it allows to structure the system in objects and at the same time to have these objects distributed along the several components of the system, making the distribution transparent to the user. JavaIDL has been chosen to implement CORBA objects. It automatically creates the skeleton and the stub starting from the IDL specification.

This work intends to show how using the MMS services and CORBA for the communication between devices makes the construction of distributed manufacturing systems easier. The creation of new VMD objects for new devices can be based on the existent objects using inheritance. An application that uses these objects does not have to worry about the communication, nor the location of the rest of elements. This has been used in order to build the VMD object for a specific manufacturing device, the RX-90 Robot by Stäubli Unimation.

2. BACKGROUNDS

In this section a brief revision of the MMS protocol and the CORBA architecture is introduced. They are the bases of the work that is mentioned in this paper.

2.1 MMS

In distributed heterogeneous systems, where each device has different features, carries out different tasks and is probably owned by a different manufacturer, the need to interconnect all devices that compound the system rises in order to achieve the integration of each one in the whole system.

Manufacturing Message Specification (MMS) is a communication language to aid the interconnection of devices in a heterogeneous environment. It is a protocol that falls in the application level standardised by ISO (International Standard Organisation).

Although MMS is not a complete Object-Oriented language, as it does not support all the features of the Object-Oriented programming, it splits the

system into objects and presents a well defined interface for each one. The most important object in the system is the Virtual Manufacturing Device (VMD). The VMD hides the real manufacturing device to the programmer. The system is structured in a set of VMDs.

MMS is also based on the client/server model. The VMD acts as a server, and so carries out a set of services that are described in the MMS protocol. The clients are the requesters of the services provided by the VMD.

A particular implementation of the MMS server must provide the mapping between the VMD model, which is an abstraction, and the functionality of the real manufacturing device.

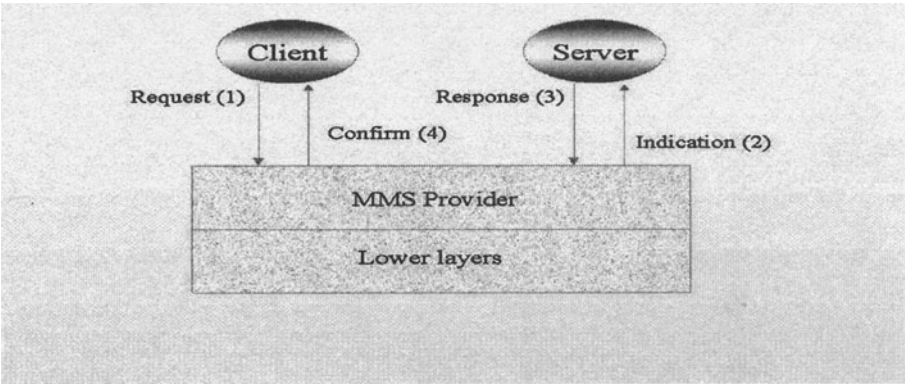

Figure 1. System based on MMS

The scheme of a system based on the communication protocol MMS is showed in figure 1.

Each MMS server manages a set of objects associated to the VMD. Some of the most important are the following:

- **The Domain Object:** It represents a subset of the capabilities of the VMD which is used for a specific purpose.
- **The Program Invocation Object:** It is a dynamic element which most closely corresponds to an execution thread in a multi-tasking environment.
- **The variable Object:** It is used to model real variables of the VMD.

Other important objects that are managed by the VMD are events. The event management services provide facilities that allow a client MMS-user to define and manage event objects at a VMD and to obtain notifications of event occurrences.

A revision of MMS can be found in [5,13] and a complete description of all these services and the protocol specification in [6,7].

2.2 CORBA

The Common Object Request Broker Architecture (CORBA) is a distributed object architecture that allows software objects to interact across networks. CORBA was first introduced in 1991 by the Object Management Group (OMG). It is an international consortium of over 800 software vendors, developers and end users.

The aim of this group is to specify an open software bus on which object components written by different vendors can interoperate regardless of the implementation language, location or host platform.

The object bus provides an Object Request Broker (ORB) that lets clients invoke methods on remote objects either statically or dynamically.

As a result of their work, OMG approved a set of specifications -called CORBA 2.0- in late 1994.

The Object Management Architecture (OMA) [10] specified by OMG, is shown in figure 2 and consists of the following elements:

- Object Request Broker (ORB): It provides the mechanisms by which objects transparently make and receive requests and responses. In doing this, the ORB provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly connects multiple object systems.
- Object Services: It provides basic operations for the logical modelling and physical storage of objects. The operations provided by Object Services are made available through the ORB.
- Domain Interfaces: They represent vertical areas that provide functionality of direct interest to end-users in particular application domains.

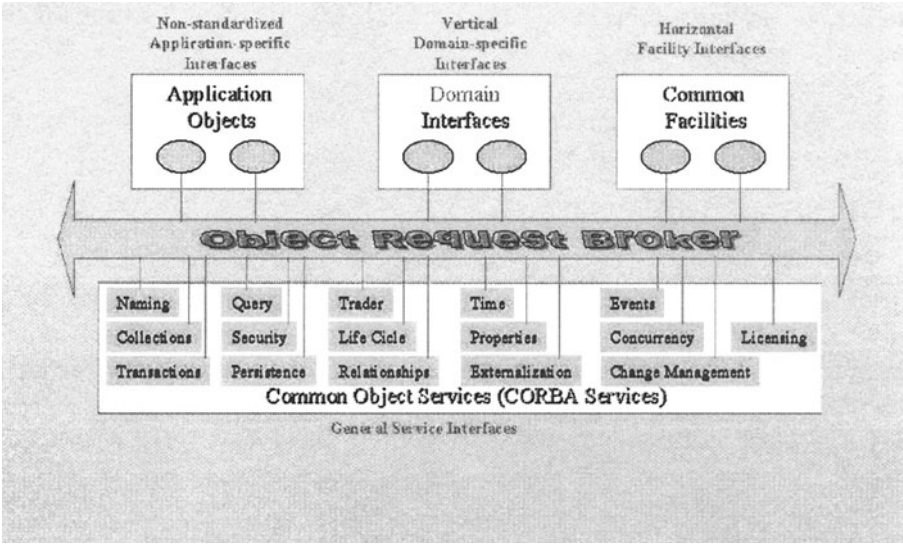

Figure 2. The Object Management Architecture (OMA) specified by OMG.

- Common Facilities: Services of direct use to application objects. It provides a set of generic application functions that can be configured to the specific requirements of a particular configuration.
- Application Objects (AO): They are specific components to end-user applications. It corresponds to the traditional notion of an application. AOs represent individual related sets of functionality.

The object specification is carried out using the Interface Definition Language (IDL). It is a descriptive language used to define the interface through which a client may access a server. IDL provides operating system and programming language independent interfaces to all the services and components that reside on a CORBA bus.

The method invocation used by the CORBA's implementation is shown in figure 3. The following components are necessary to carry out the invocation:

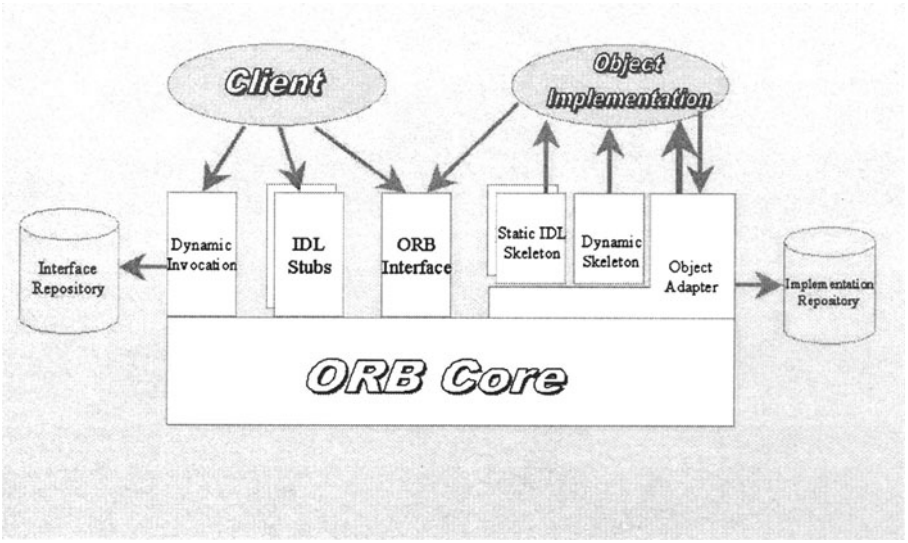

Figure 3. Object's method's invocation.

- Client IDL Stub: Client code used by an object to encode invocations in a form which can be handled by the ORB, and to decode replies received via the ORB.
- Skeleton: Server code up-called by the ORB, capable of decoding requests transmitted by the ORB, converting it into an invocation of the implementation object, and encoding the results to be sent back to the client via the ORB.
- Object Adapter: It defines how an object is activated. It can do this by creating a new process, creating a new thread within an existing process, or by reusing an existing thread or process.

More information regarding CORBA can be found in [10,11,12] and about CORBA applied to manufacturing systems in [4,9].

3. AIM OF THE WORK

The objective of this work is to implement the VMD object for the RX-90 robot. However, the development has consisted in two phases. In the first one, a generic VMD has been carried out conforming to ISO/IEC 9506 part 1 [6] and part 2 [7]. This generic VMD (presented in [2,3]) can be inherited by other classes in a natural way. Doing this, the construction of a specific

VMD is more straightforward and it is suitable not only to build the VMD for the RX-90 but for whatever manufacturing device that it is necessary.

Figure 4. RX-90 Robot

The specific VMD for the RX-90 is built starting with the generic VMD, conforming to ISO/IEC 9506 part 3 standard [8], taking into account that this implementation must be used as the base to build other specific VMD's for robots making the least number of changes possible.

The prototype includes the parts concerning the VMD, Domains, Program Invocations, variables and events (this last one, only in the generic VMD). It does not include Semaphore, Operator Station, or Journal objects, but the system could be extended to them.

The main objective is to build classes that can be used as the base for the implementation of VMD objects that represents other different devices. In addition to this, a MMS client has been developed, which can be used in the future in order to build control distributed systems.

4. SYSTEM DESCRIPTION

The scheme of the system architecture is shown in figure 5. As the standard describes, the MMS client also carries out MMS services, its behaviour is as a server from this point of view.

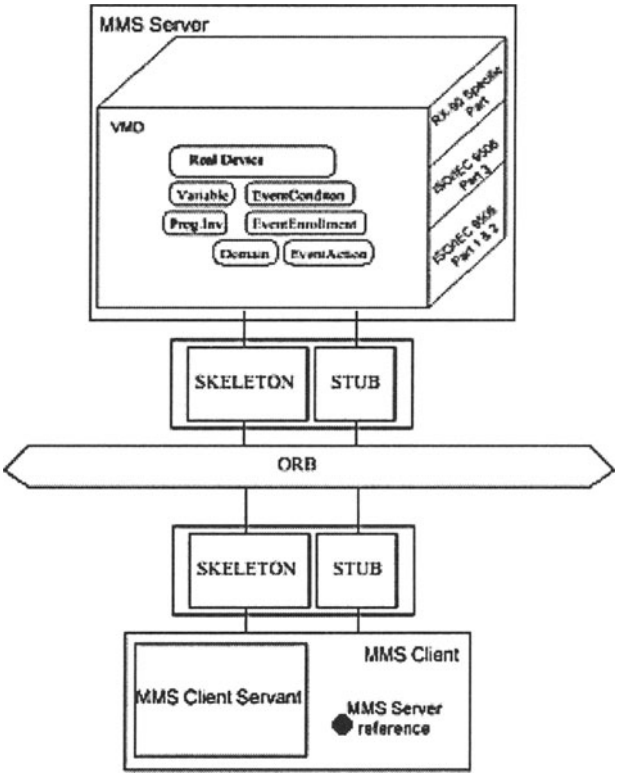

Figure 5. System Architecture

On the server side the components are the following:

- MMS Server: CORBA object that can be accessed through the object bus. The services provided by this object are the services specified in the MMS standard for the server. The description of this object is specified using the Interface Definition Language (IDL).
- VMD: the MMS Server delegates in the VMD to resolve the MMS services.
- Skeleton: It allows the clients to call the MMS Server methods using the ORB.
- Stub: It is used in order to carry out the remote invocation to the methods of the MMS Client Servant object so that the server can notify the events to the client when they happen and carry out requests of MMS services that the client can accomplish.
- Real Device: It is the physical device that is hidden by the MMS Server object. In this work the RX-90 [1] device is used.

On the client side, the following components can be identified:

- MMS Client: It is responsible for calling the services using CORBA. Its task is to get the server references and to register the client servant in the ORB.
- MMS Client Servant: It carries out the client services and receives requests from the MMS servers.
- Skeleton: It allows the MMS Client Servant object methods to be called through the ORB.
- Stub: It undertakes the responsibility for calling the remote methods of the MMS Server object.

The VMD is divided in three different layers, which are the following:

- Generic VMD: It corresponds to ISO/IEC 9506 parts 1 & 2 that include the general services of the VMD.
- Robot VMD: It corresponds to ISO/IEC 9506 part 3 that include the companion standard for robots.
- RX-90 specific part: This is the layer that depends on the specific device.

5. IMPLEMENTATION

A prototype for this work has been built, where the programming language JAVA has been used. JAVA language has the following features:

- Object Oriented programming language: It has the benefits of this methodology (reusability, facility in the integration, debugging and maintenance). Throughout, e.g. no coding outside of class definitions, including main(). An extensive built-in class library.
- Familiarity: It is similar enough to C and C++ that experienced programmers can get going quickly.
- Simpler than C & C++: Because it has no pointers, no preprocessor and automatic garbage collection
- Portability: Code is compiled to bytecodes, which are interpreted by the JAVA virtual machine.
- Robustness: Exception handling built-in, strong type checking (i.e. all variables must be given explicit type), local variables must be initialised.
- Threading: Lightweight processes, called threads, can easily be spun off to perform multi-processing.
- Dynamic Binding: Even if libraries are recompiled, there is no need to recompile the code that calls classes in those libraries since binding, i.e.

the linking of variables and methods to where they are located, is done at runtime.
- Platform Independence: The JAVA Virtual Machine is available in many types of computers and OS's. The Code that can be exchanged without requiring rewrites and recompilation would save time and effort.
- Security: No memory pointers exist. Programs run inside the virtual machine sandbox. The code is checked for pathologies by the bytecode verifier, the class loader and the security manager.

In order to communicate with the real device the javax.comm library has been used. The prototype can be run in any computer and operating system where the Java interpreter can be run. The VMD used is a CORBA object developed in [2] that attends to MMS requirements with extensions for the robot.

CORBA has been used as the communication architecture and the ORB chosen is JAVAIDL because previous work had been done with it.

A client with the basic functionality has also been implemented. It allows the operator to load and run programs over the robot, show variable values, etc.

6. CONCLUSION

In this work, the VMD for the RX-90 robot over CORBA and a basic client that allows access to the MMS services have been implemented. With these components, a set of functions can be carried out over the robot, as load and run programs in a remote way, create, read and write variables, and so on.

Reusability and modularity are features achieved in this development. It allows having a base to implement new clients using inheritance. On the other hand, using CORBA, as the communication platform to accomplish the interaction between clients and servers, facilitates the creation of control distributed systems.

7. ACKNOWLEDGEMENT

This work is supported in part buy the CICYT under grant num. TAP-98-0541

8. REFERENCES

[1] Adept Technology, Inc, "V+ Language Reference Guide", Part Number 00961-00100,Rev. B. Version 11.3T. July 1996.

[2] T. Ariza, F.R. Rubio, "Communicating MMS Events in a Distributed Manufacturing System using CORBA", Preprints DCCS'98,1998.

[3] T. Ariza, F.R. Rubio, "MMS-Manager: Device Management in Heterogeneous Environment Based on CORBA", Preprints Controlo'98,1998.

[4] Carvalho, A.S. and M.J. De Sousa. "Development of an ORB for Distributed Manufacturing Applications", WFCS'97 Workshop (1997).

[5] CCE-CNMA, ESPRIT Consortium. "MMS: A Communication Language for Manufacturing", Springer, 1995.

[6] ISO/IEC 9506-1. "Industrial Automation Systems- Manufacturing Message Specification", Part 1: Sevice Definition. International Standars Organization, 1990.

[7] ISO/IEC 9506-2. "Industrial Automation Systems- Manufacturing Message Specification", Part 2: Protocol Specification. International Standards Organization, 1993.

[8] ISO/IEC 9506-3. "Industrial Automation Systems- Manufacturing Message Specification", Part 3: Companion Standard for Robotics. International Standards Organization, 1993.

[9] P. Newmann, F. Iwanitz. "Integration of Fieldbus Systems into Distributed Object-Oriented Systems", WFCS'97 Workshop (1997).

[10] Object Management Group. "CORBA: Common Object Request Broker Architecture and Specification", Published by the Object Management Group (OMG), Framingham, MA. 1995.

[11] Object Management Group. "CORBA services: Common Object Services Specification", Published by the Object Management Group (OMG), Framingham, MA. 1995.

[12] R. Orfali, D. Harkey, J. Edwards. "The Essential Distributed Objects. Survival Guide", Wiley, 1996.

[13] Pimentel, Juan R. "Communication Networks for Manufacturing". Prentice Hall, 1990.