A FRAMEWORK BASED ON CORBA AND OO TECHNOLOGIES FOR REMOTE ACCESS TO INDUSTRIAL PLANTS

Isidro Calvo, Marga Marcos, Dario Orive

Department of Automatic Control and System Engineering
E.T.S.I. de Bilbao (University of the Basque Country)
Alda Urquijo, s/n, Bilbao (SPAIN)
fax: +34 946 01 41 87
e-mail: {jtpcagoi, jtpmamum, jtporred}@bi.ehu.es

Keywords: Remote control. Distributed computer control systems. Object modelling techniques. Flexible manufacturing systems. Computer-integrated manufacturing.

Abstract

The current paper presents an overview of a general architecture that provides remote access to industrial plants. In this paper the components of the architecture are described in detail, focusing on the key element of the architecture, which is the Application Server. This architecture has been applied to a case study consisting of a manufacturing cell to provide users located anywhere on the Internet remote access to the system under control.

1 Introduction

Today, the interconnection of field area devices to the Internet is one of the most challenging topics in automation. There are several reasons that make this interconnection worth studying. On one side, the worldwide acceptation of Internet as a communications medium has converted it in a "de facto" standard, used even internally in proprietary networks (intranets). Moreover, modern information technologies make possible the integration of control systems in distributed environments connected by high-speed networks, such as fieldbuses, Ethernet, ATM and wireless networks. Even though, at the moment a lot of users do not trust this kind of technologies to be applied to industrial fields, in our opinion they are mature enough to be used in this kind of environments, allowing remote users to directly monitor and teleoperate control systems or to integrate the field area devices in higher-level applications.

Obviously, the integration of all the necessary components, in order to obtain a distributed application working in an Internet type environment, is not an easy task. In [1,2,4] it is presented a methodology that provides a framework to build this kind of systems. This methodology is based on object-oriented architectures that adapt remarkably well to Internet environments such as CORBA [3], DCOM or Java/RMI. These architectures permit to use remote objects as if they were local and provide interesting services that ease the development of applications. In [6] it is possible to find a comparative among them. Moreover, the use of Java language provides interesting characteristics such as platform independence and easy integration in HTML browsers. Besides, using object-oriented technologies also achieves the typical advantages provided by those (such as objects reusability or easier maintenance).

This paper details the internal architecture of the key element of the architecture, the application server, which is the gateway that provides remote access to remote users. It also describes how the whole architecture has been applied to a case study composed by a manufacturing cell.

The layout of the paper is as follows: section 2 summarizes the proposed architecture as it has been presented in previous papers. Section 3 details the generic architecture obtained for the application server and the methodology used to obtain it. It follows another section with the application of the whole architecture to a case study. Finally, the last section draws some conclusions.

2 General architecture overview

This section presents the architecture proposed. This architecture pretends to be generic enough to be used with different kinds of industrial plants. However, some requirements need to be satisfied. For example, it is convenient that plants behave autonomously. This is due to the fact that, at the moment, Internet connections are not reliable enough to make systems that respond adequately to time requirements. This autonomous behaviour, nevertheless, is frequently found in industrial processes.

Broadly, our architecture divides systems into the following categories (see Fig. 1): physical plants, plant controllers, application servers and remote client applications.

- ***Physical plants*** are made out of the connected field devices. These may consist of several devices connected to any kind of local network, fieldbus, etc, or directly to the plant controller. In fact, physical plants may be seen as the process under control.
- ***Plant controllers*** take care of the local behaviour of the applications, as well as of providing application servers the information they need. They may also provide user interfaces for specific local operators who take local decisions over the system under control. These components may be either designed ad hoc or a legacy controllers (CORBA-like architectures may help here).
- ***Application servers*** act as gateways that connect the local controllers to the Internet. It is desirable that these nodes are connected to plant controllers with dedicated links, so they may exchange frequently information about the state of plants. They are the key elements of the architecture as they contain a model of the plant under control, which it will be called from here on the ***Virtual Plant***. Virtual Plants consists of a set of objects that contain the relevant information as well as the methods with the operations remote users may perform over the plant. They are also responsible for the distribution of the alarms. Other tasks are related to the users management: remote users validation, filtering the operations, etc. In short, they offer a CORBA interface to remote applications.
- ***Remote applications*** allow remote users to operate the system. They will use CORBA objects that communicate with the CORBA interface provided by the application server. They will be responsible for presenting the information to remote users. Here, the use of Java applets together to the use of Internet browsers (Internet Explorer, Netscape…) provides a standard interface that eases the application learning curve.

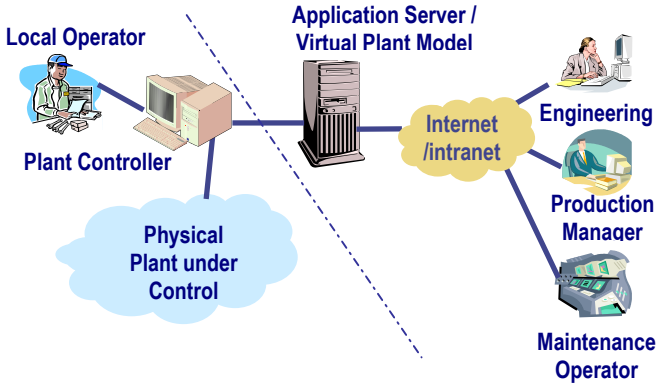

Fig. 1. General architecture overview

It is important to remark that this architecture allows remote applications to integrate easily information obtained from several application servers (each of them containing objects that model several physical plants), as well as to provide remote access to the physical plants from anywhere in the Internet through the application servers. Besides, the use of distributed object architectures such as CORBA permits to divide the application between several application servers.

This architecture may be complemented with other nodes such as databases Web servers, etc.

In order to filter the operations that may be undertaken over the system it will be convenient to define several user profiles that will group all operations a specific kind of user may take over the system. This will be explained in further detail later.

This approach allows remote users to perform several kinds of operations over the system: These operations may involve

- Monitoring
- Information requirement
- Configuration
- Orders over the physical plant
- Alarm distribution

3 Architecture of the application server

3.1 Methodology used to design the application server

According to the approach presented in [5], there are three main steps in the design of a distributed architecture for a system. Figure 2 depicts these three basic stages:

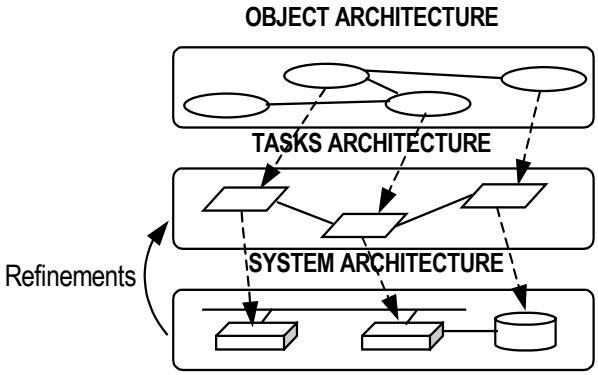

Fig 2. Stages in the design of a system

- ***Object architecture*****:** It will consist of identifying a collection of objects that communicate and collaborate to implement a specific function.
- ***Task architecture*****:** This stage will consist of identifying the concurrent tasks in which the system will be divided in order to perform the desired behaviour.
- ***System architecture*****:** Here, it will be detailed how the system is assembled from a set of building blocks, by selecting from among various computational elements and topologies. This stage will include the identification of the nodes in which the system is implemented. At this

stage, depending on the tools (operating systems, programming languages… etc) may be necessary to perform several refinements.

Next, the architecture of the application server will be described in these terms.

3.2 Object architecture

This stage identifies the objects that compound the application server. This task was mainly presented in [1,2]. Fig 3 shows the main components found at this stage. Following there is a brief description about the functionality of each of them.

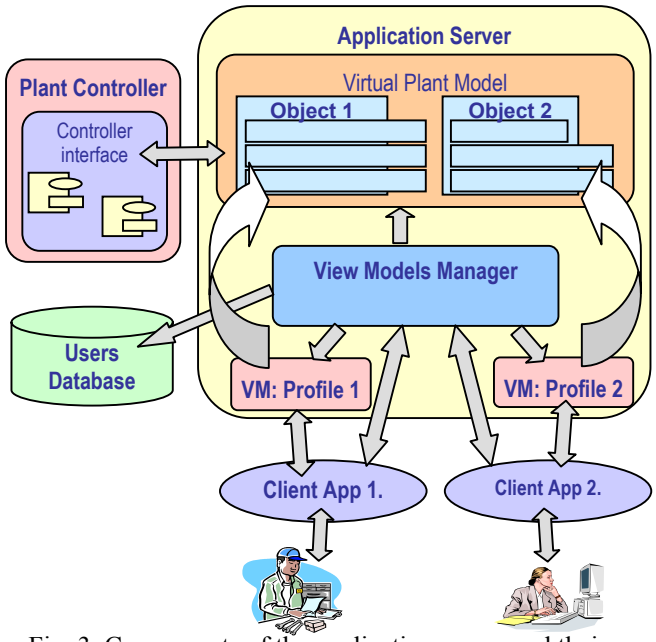

Fig. 3. Components of the application server and their relation with other components of the architecture

- ***Virtual plant***: It contains a collection of objects with information and actions relevant to remote users. Evidently, these objects may be reused in other applications, which would reduce the efforts in the building of new applications. These objects cannot be accessed directly by remote users, instead, they are protected by the View Models that will deliver the orders received from the remote users.
- ***View Models Manager***: This a single object responsible for offering the management task of the system. Some of these tasks are authentication of remote users and distribution of alarms that may be originated by the plant controller or by other objects of the Application Server.
- ***View Models***: Different remote users are assigned to a specific profile. Profiles allow the application server to filter the operations that may be performed over the system. The view models will be the objects that correspond to every profile. These objects dialogue with all the remote users of a specific profile.

3.3 Task architecture

This section describes which tasks will perform every object identified in the object architecture.

3.3.1 Virtual Plant

The virtual plant is just a container of objects, so at this stage, it will be necessary to identify the actions that will perform these objects. These tasks involve the continuous updating of information obtained from the plant controller as well as the actions required by remote users through the corresponding View Models. Upgrading the information inside these objects is an internal task performed periodically depending on the period set up to refresh every virtual plant component.

3.3.2 View Models Manager

This object will perform two main tasks: allow remote users to open new connections and alarms distribution.

3.3.3 View Models

They concentrate the connections between the application server and all remote users of a specific profile. This connection involves the following tasks:

- Accepting orders from remote users that will be processed by the methods of the virtual plant objects.
- Delivering monitoring data to remote users.
- Delivering the alarms between all the remote users of the profile.
- Acknowledging the alarms.

For some of these tasks it will be used the CORBA event service [7]. The event service defines several models depending on the desired behaviour of the application. Briefly, all models use an event channel that is used to deliver some data to several users. Two of these models are the PUSH-PULL model and the PUSH-PUSH model. In the first model it is the supplier who sends the data to the event channel where it remains until a remote user recovers it. The second model, the PUSH-PUSH model, is useful to distribute the same information to several registered remote users. In this case, the supplier sends the data to the event channel and this one distributes it between all the registered users.

Figure 4 shows the task architecture for a View Model of one user profile. In the figure three remote clients of the same profile are connected to the View Model.

The monitoring task thread uses the CORBA event service. This thread is responsible for obtaining the data from the Virtual Plant objects and sending it to the Monitoring Event Channel. In this case the most suitable model is the PUSH-PULL. This procedure will allow remote users to specify the frequency at which data will be recovered from the channel.

This ability of setting up the frequency is very interesting in Internet like environments to adapt to the QoS requirements.

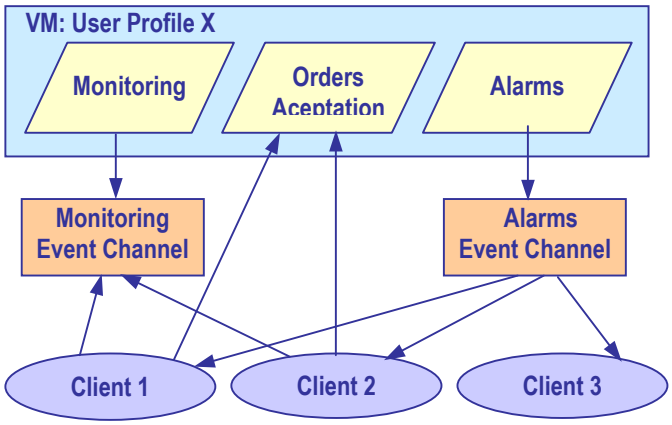

Fig. 4. Task architecture for View Models

Orders are processed by the Orders acceptation thread. All operations that are required by remote users will pass through this thread.

Finally, the alarm delivery is undertaken by another thread that receives the alarms from the View Models Manager and after logging the alarms it sends them to the corresponding Alarm Event Channel, and this one delivers the alarms among all the registered users. In this case the PUSH-PUSH CORBA event model is the one that adapts best to the required performance. Once remote clients receive the alarms, any of them may acknowledge the alarm and take the corresponding actions over the plant to counteract the alarm.

3.4 System architecture

Finally, it will be also necessary to distribute the different objects among the nodes that will constitute the system.

This stage is highly dependent on the specific requirements for the system, such as operating system, computation power needs, etc.

3.5 Security

It is clear that in an architecture that offers access to remote users through Internet security is a key point. This requirement will by achieved by implementing the currently available technologies in the application server.

In [9] it is possible to find a discussion about the available techniques. These techniques provide:

- Confidentiality
- Authentication
- Integrity
- Non-repudiation

These techniques are based on cryptography. Broadly, there are two kinds cryptographic techniques; symmetric key encryption and public key encryption. The first one uses the same key for both the sender and receiver of the information, (examples of these algorithms are DES or RC5), whereas the public key encryption use different keys that must match. The second type of encryption is much securer but it also requires much more processing power, slowing the applications. This is why they are usually employed together.

In our framework both techniques will be used. The monitoring and alarm information will be coded before it is sent to the corresponding event channel with a symmetric key algorithm being the clients responsible for the decoding. The symmetric key both ends will use during the connection must be delivered at the beginning of the connection with the view models manager by using public key encryption techniques. In order to check that messages are not altered during the delivery hash functions must be also used to test their integrity.

On the other side, orders undertaken over the application server must be signed by the remote user's digital signature. This method will provide both authentication and non-repudiation. However, it will require the use of a third party responsible for certifying the digital signatures.

These techniques have proven reliable enough depending on the length of the keys (this is a critical parameter above all in symmetric key techniques). In this scheme key management, above all, during public keys delivery, is a crucial factor as here is where most attacks are usually directed. In [8] is proposed a method to improve this key delivery.

4 Case Study

In order to validate the architecture proposed above, this section presents the implementation of these ideas in a real case study composed by a manufacturing cell.

4.1 Physical plant

The physical plant will consist of the following devices attached to a MAP-Ethernet network with MMS protocols as it is shown in figure 5.

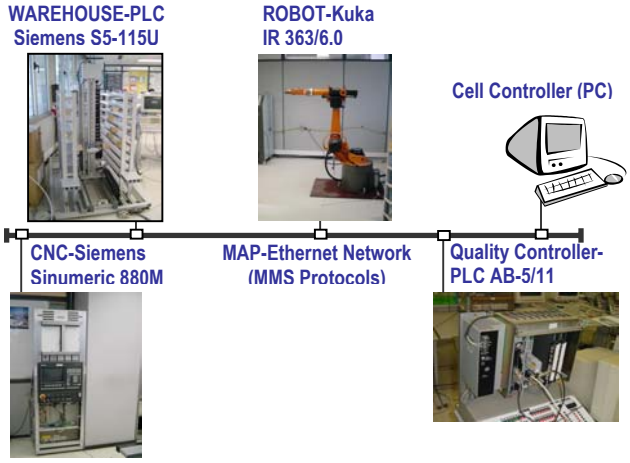

Fig. 5. Physical plant devices of the case study

- ***Warehouse***: It is responsible for storing both the raw and processed material. This device will be controlled by a PLC that accepts the orders from the cell controller.
- ***CNC***: It manufactures the raw items withdrawn from the warehouse. The cell controller will supply the necessary programs used to manufacture the items.
- ***Robot***: It assumes the role of being the transport element of the cell. The programs that define the movements of the robot will be also downloaded by the cell controller.
- ***Quality controller***: It consist of a PLC responsible for identifying those items that do not pass a quality test in order to discard them.

In this case study, the communications among the devices are provided by a MAP-Ethernet network because being itself object oriented it eases the task of identifying the objects of the virtual plant. However, it will be valid any other network.

4.2 Cell Controller

The cell controller is another node attached to the manufacturing network, as figure 5 shows. This node takes the local decisions over the plant and it is relatively autonomous.

In this case study, the cell controller has been developed ad hoc, nevertheless, CORBA may be also used to provide an interface to a legacy cell controller.

Local operation applications are allowed to perform operations through the CORBA interface that are not permitted to remote users. In most cases this is due to that some operations must be performed at the physical plant site.

4.3 Identified profiles

For this case study, the following profiles have been identified. More information about every user and the operations they are allowed to may be found in [1].

- Cell operator
- Maintenance operator
- Warehouse manager
- Engineering
- Production manager
- Client
- Users Administrator

4.4 Application Server

4.4.1 Object architecture

According to the framework presented in the previous section, the object architecture in this case will be composed of the virtual plant, with all its objects, the view models manager, and a view model for every profile described above.

The most interesting part of this section is the virtual plant. Objects inside have been described in [1]. Figure 6 presents a class diagram for the case study.

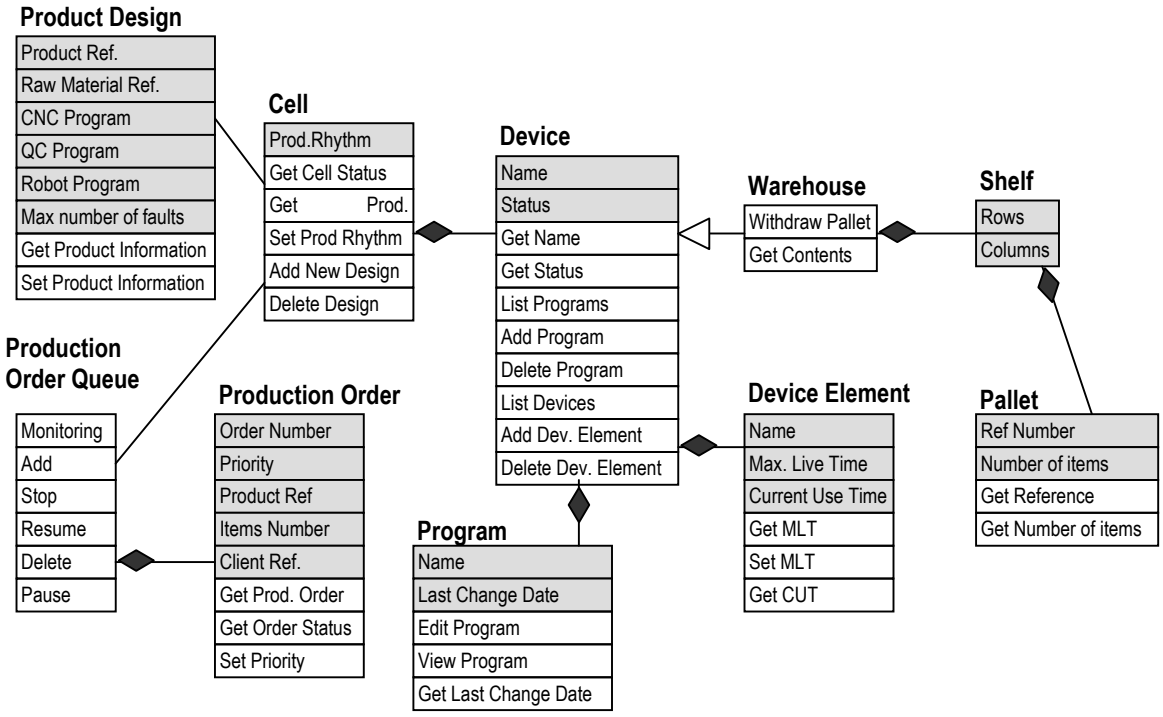

Fig. 6. Class diagram for the objects contained in the Virtual Plant

4.4.2 Task architecture

As described in section 3.3 objects in the virtual plant will request the information from the physical plant periodically. The view models manager will consist of two threads, one for accepting connection orders and the other for distributing the alarms. Also there is one view model for every profile that will follow the structure described above.

4.5 System architecture

Figure 7 depicts the architecture used for this case study. This diagram may be compared with figure 1.

The cell controller node has been implemented on a Pentium III running Windows NT 4.0. The cell controller was implemented under Visual C++ and the SISCO MMS-Ease libraries. The interface between the cell controller and the application server was completed with CORBA (Visibroker).

The application server has been implemented over a Pentium IV running Windows 2000. The application server was implemented with Java and CORBA (Visibroker).

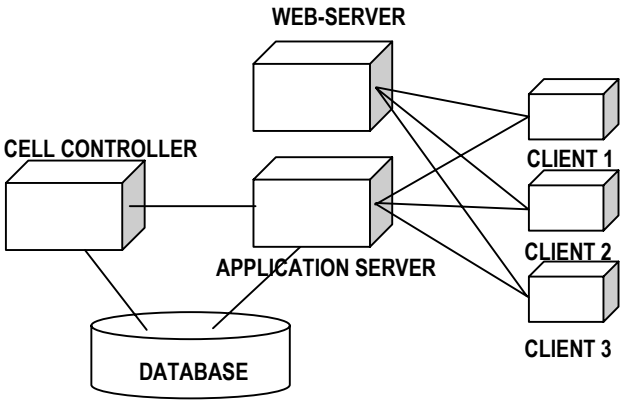

Fig. 7. System architecture for the case study

Another node holds the database used to keep all the production historic files, as well as the users database, etc. This database has been implemented using MySQL. This database will keep process information, the users database, the historic files, alarm historic files…etc..

Finally, a Web-server is used to download the CORBA-Java applets remote users will use. This eases the distribution of the applications.

Remote clients will use a navigator, such as IE, Netscape or any other to access the application server. Remote nodes may be running any operating system, as long as they have the Java virtual machine installed and there is a navigator for this operating system.

5 Conclusions and Future Work

The architecture presented in this paper allows remote access to industrial plants. It pretends to be generic enough to be used with industrial plants from different environments, as long as they present a relatively autonomous behaviour. It has been implemented over a case study consisting of a manufacturing cell.

In this sense, the next step will be to investigate the features of RT-CORBA in order to overcome the response time limitations of CORBA in such type of applications.

Acknowledgements

This work has been supported by CICYT project DPI2002-03946

References

[1] Calvo I., M. Marcos, I. Sarachaga, D. Orive, "Using OO Technologies in Factory Automation" *Proc. of the 28th IECON Conference* Seville. (2002)

[2] Calvo I., M. Marcos, I. Sarachaga, D. Orive, "Using UML for modelling remote access to manufacturing systems" *Proc. of the 15th IFAC Congress* Barcelona. (2002)

[3] Henning M. and S. Vinoski "*Advanced CORBA Programming with C++*", Addison Wesley Longman, Inc. (1999)

[4] Marcos M., J.M. Fuertes, I. Calvo, P. Martí, D. Orive, R. Villá, I. Sarachaga and S. Buzoianu "Object-Oriented Modeling for Remote Monitoring of Manufacturing Processes" *8th IEEE International Conference on Emerging Technologies and Factory Automation*, Nice. (2001)

[5] Moore A., Cooling N. "Developing Real-Time Systems using Object Technology – Overview". URL: http://www.artisan.com (2000)

[6] Orfali R., D. Harkey (1998) *"Client/Server Programming with Java and CORBA"* John Wiley & Sons. (1998)

[7] Object Management Group, *Event Service Specification* URL: http://cgi.omg.org/docs/formal/01-03-01.pdf (2001)

[8] Schwaiger C., Sauter T., "A Secure Architecture for Fieldbus/Internet Gateways" *8th IEEE International Conference on Emerging Technologies and Factory Automation*, Nice. (2001)

[9] Weaver A C."Privacy and Security on the Internet", URL: http://intercom.virginia.edu/crypto/privacy_and_security.pdf

[10] Wollschlaeger M "Framework for Web Integration of Factory Communication Systems" *8th IEEE International Conference on Emerging Technologies and Factory Automation*, Nice. (2001)