CORBA for Distributed Measurements

Enric TRULLOLS [1], Carlos SAMITIER [2], Jordi SORRIBAS [3], Antoni MANUEL [1], Rafael MORILLAS [1], Joaquin del RIO [1]

[1] Technical University of Catalonia.
Av. Victor Balaguer s/n. 08800 Vilanova i la Geltrú, Spain

[2] Global Networking Enginneering,
Fòrum Nord, Marie Curie s/n. 08003 Barcelona, Spain

[3] UTM-CMIMA,
Psg. Maritim de la Barceloneta 37-49. 08042 Barcelona, Spain

ABSTRACT

The idea of a distributed computing environment has been around for a long time. In the last decade computing centers have survived serving users remotely via Internet while maintaining essentially a mainframe and terminal model. Virtual instruments and advances in networking technology make it possible to implement modular distributed measurements and computing systems. The success of "SETI at home" project is a good example of distributed computing systems. Sharing resources and the elimination of obstacles introduced by communication networks could be the key to open the doors to a new measurement philosophy, in which virtual instruments will play an important role. Thanks to this working principle, it is possible to correlate data from different experiments or complement one experiment with information gathered from the local environment or any other remote site.

This paper also describes the actual status of the LabVir project that we are carrying out in order to model and implement distributed research in the Spanish oceanographic vessels. User interfaces based on Web technology, will allow friendly accessibility from any place with HTTP connectivity. Experiments tools will be accessible using any kind of the available internet browsers.

Keywords: LabVir, CORBA, Java, UML, XML mobile agents, marine research.

1. INTRODUCTION

Oceanographic vessels are real floating labs with a lot of instruments and computers associated to them. Because there is not a big market for specific scientific marine instrumentation, some devices are far from standardization and must be regarded as prototypes. That usually means we are working simultaneously with several hardware and software platforms and having serious problems to interconnect them [1], [2]. Same data, as temperature, wind and water speed, bottom depth, etc., are collected automatically by the different instruments and sensors on board. Each instrument has an associated computer that stores data locally. Usually instruments are physically located in different places and need to interchange information in real time. In order to prevent chaos due to the physical dispersion of computers on board, a centralized control process is needed. A single, centralized, full register of all data (apart from local data stored in the different acquisition computers) is also required.

Specific scientific tasks on board require visualization of data in real time, access to historical data and integration of data coming from other sources, including ground labs [3], [4]. Satellite remote access to the acquisition system made it possible to correlate data from different experiments or complement one experiment with information gathered from the local environment or any other remote site.

In order to achieve the smart and modular features in the measurement systems, we need to define an architecture in which the systems could interoperate in heterogeneous platforms. In the above described scenario we are designing and implementing a system orientated towards distributed objects with Web accessibility using UML. The instruments and applications distributed in the vessels will be accessible from any executed application in a server, as if they would be physically situated in the same machine. Since Spanish oceanographic vessels work in remote areas, as Antarctica, for a significant period of time, Web accessibility is regarded as a true value.

2. SYSTEM DESIGN. UML

Water temperature (for example) is an irrelevant data if you do not know the collecting time, the geographical coordinates and other parameters associated to them. Because of that, acquisition system must be modeled as a whole system and have to take into account the characteristics and relations between the different entities. If not, the system will be composed by a set of programs unable to interoperate and store

data in a structured way, and consequently with a poor capacity to offer a useful and global vision to the final users.

The complete acquisition system is modeled with UML (Unified Modeling Language) [5]. Originally created to model and document orientated object applications, UML is a powerful tool to analyze any kind of system in which we can identify its components, expressed as entities or classes, and the logical relationship between them. Using UML it is possible to describe all parts of the distributed system no matter it nature (PC, workstation, intelligent sensor, electronic system, etc.). In addition, UML design is independent of the implementation language.

Basic requirements of the acquisition system are shown in the UML use-case diagram (figure 1). In it, you can notice that

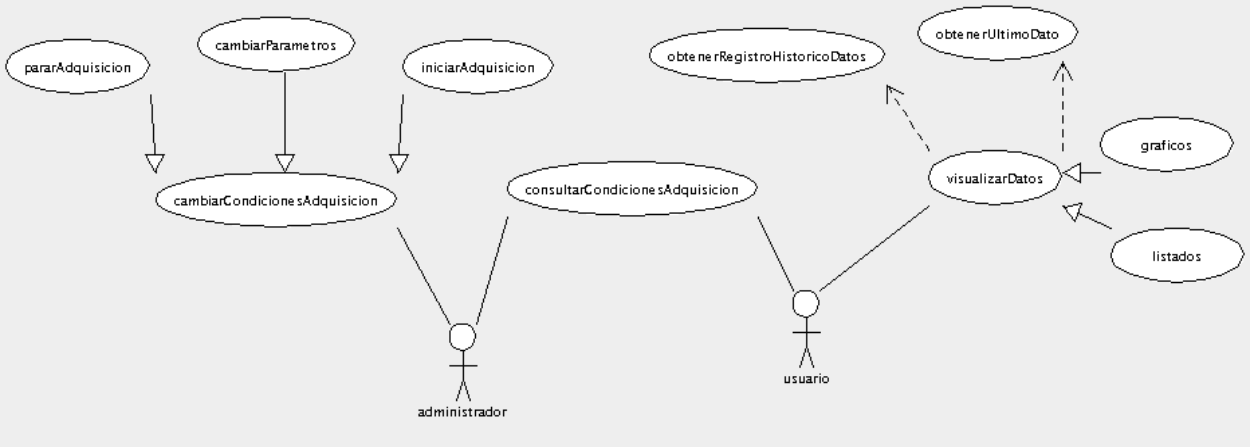

Figure 1. UML use-case diagram

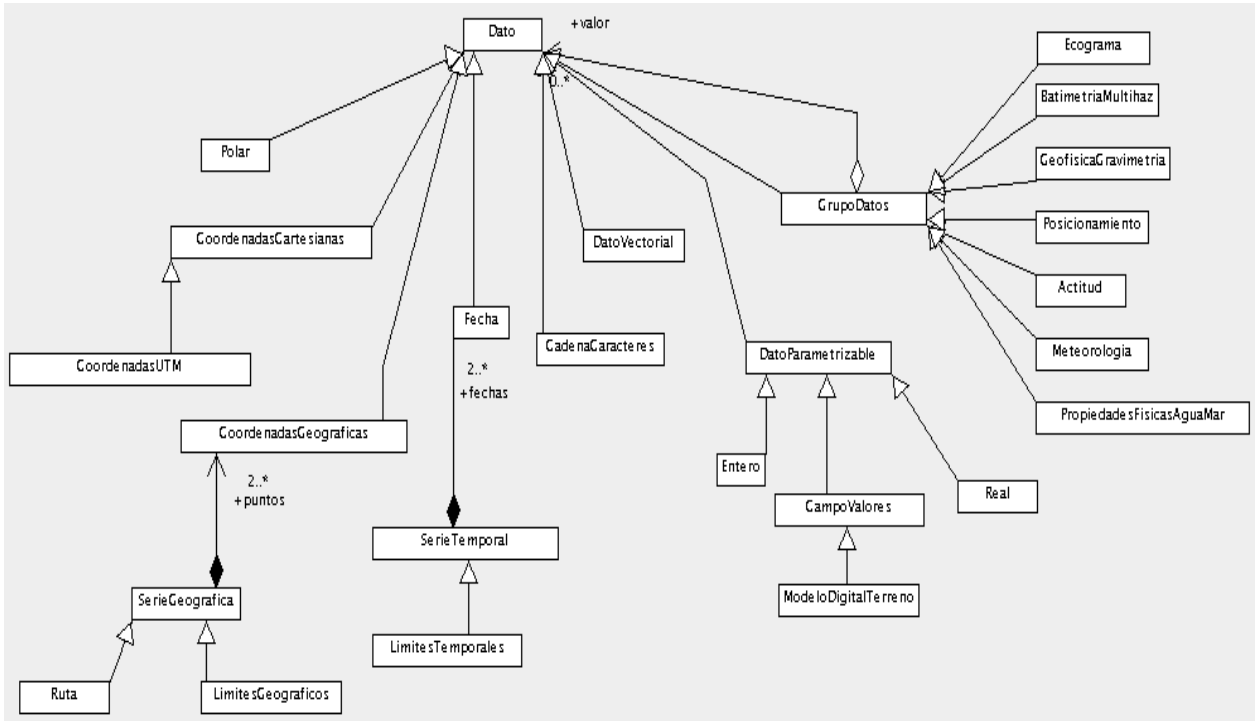

Figure 2. Class diagram for data entities

although the main task is the information acquisition, real/differed time access is one of the user requirements. In order to model a generic acquisition system we have identified three different entities:

- Data. Information gathered from sensors and instruments on board. Also complementary information needed to understand scientific results, as calibration parameters and geographical-temporal coordinates.

- Logical elements. Programs and software elements involved in acquisition, storage, visualization and representation of data.

- Physical elements. Hardware, electronics, equipment and sensors associated to acquisition process. By extension, the system manager, the users and the entire acquisition platform on board.

The three entities (data, logical elements and physical elements) have been modeled using UML. UML (and the abstraction associated to it) must be regarded as a powerful tool to define system functionality, and not only as a documentation procedure [6]. Figure 2 show the UML class diagram corresponding to the data entities. Logical and physical element class diagrams (not shown is this paper) can be connected to data diagram using heritage, association or use links.

3. INVOLVED TECHNOLOGIES

Diagram in figure 3 outlines the different techologies involved in LabVir project.

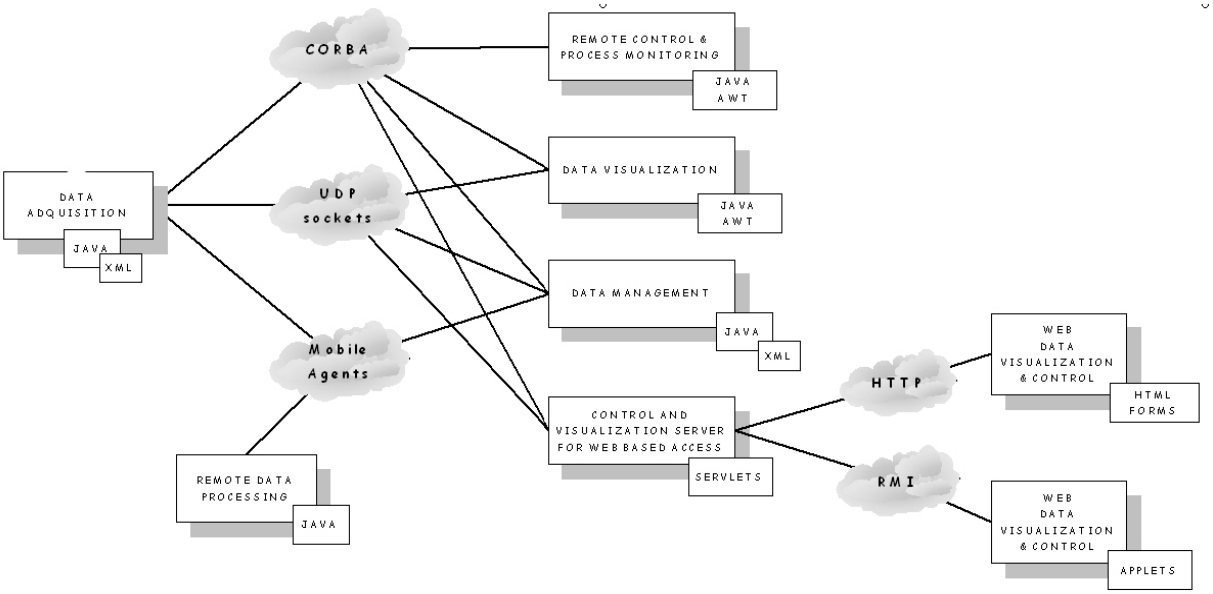

Figure 3. Technologies used in LabVir Distributed Environment

3.1 Multiplatform and code portability. Java.

We have selected Java as an implementation language because it allows developping portable applications between different hardware platforms in a global way. Java also permites to reuse the class libreries to process and represent Scientific data.

Java Virtual Machine (JVM) abstracts applications from operating system and allows to develop all the applications in this language. Java, as an interpreted language, is independent of the platform where it runs. In each platform, we can have a specific JVM, which will interpret our Java code. For exemple we have experimented with a TINI embedded system environment taking advantage of its reduced dimensions and

its low cost. Our TINI has a JVM, which will be executed in its 8-bit microcontroller. The use of Java in small embedded systems is a reality and these systems must be regarded as a part of a distributed acquisition system [7].

Finally, all the implemented applications could be executed on Microsoft Windows, Linux, UNIX, or under TINI environment in a fully transparent way (figure 4).

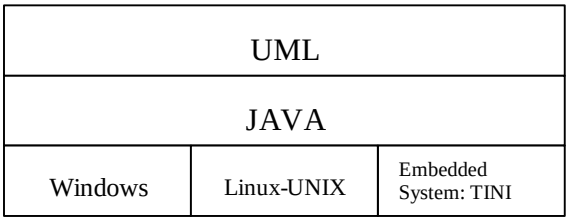

Figure 4: Lenguages and technologies.

3.2 Data portability. XML.

Due to the large amount of scientific topics and parameters covered in marine sciences, oceanographic data has been represented by very different data formats and structures, that has been adopted following the data acquisition storage and processing requirements. In many cases each equipment has its own data format that makes data sharing and also comparison between different scientific teams difficult, forcing to the format standardization. In this sense many initiatives has been appeared, like netCDF or HDF file formats that allow accomplish data auto description and portability between hardware platforms, nerveless this kind of files need software libraries to be interpreted.

In LabVir project we have adopted XML (eXtended Markup Languaje) to represent data in all data input/output procedures. This language, similar to HTML, was originaly designed to describe and trasport data in ASCII format, being an independent mechanism of software and hardware platforms. XML is an excellent language for defining data structures, providing good jerarquical relation between them, and storing not only data but metadata too. Because it is based on an ASCII representation XML ensure data portability, which is a requirement in distributed systems like ours. International oceanographic community is also working on a extension of XML called Marine XML [8].

XML not only is used to represent data, but we use it in oftware and hardware device configuration files too. XML rovide external mechanisms for validating and converting data o other graphic or publishing formats, getting away these tasks from the code.

Despite XML files are ASCII based, we can store binary data as NIME or Base64 into them as binary attached files in basic e-mail system. As Labvir is fully implemented in Java, we have used several Java Classes to create, transform and read XML files in an easy way, as JDOM or Castor libraries.

XML does not define any protocol or especific mechanism for data transmition. In the present application, we are using IIOP (Internet Inter-ORB Protocol). Following boxes show two examples of XML code: 1) a device XML configuration file, and 2) a XML file for ship positioning data.

```
<?xml version="1.0" encoding="UTF-8"?>
<GeneradorEventos>
    <nombre>Disparador Sismica</nombre>
        <DispositivoTemporizadorMultihilo>
            <nombre>Trigger</nombre>
            <frecuencia>5000</frecuencia>
        </DispositivoTemporizadorMultihilo>
        <DispositivoUDPMultihilo>
            <nombre>Disparador</nombre>
            <puerto>5000</puerto>
        </DispositivoUDPMultihilo>
        <DispositivoSerieMultihilo>
            <nombre>GPS</nombre>
            <puerto>/dev/ttyS0</puerto>
            <baudios>4800</baudios>
            <bitsDatos>8</bitsDatos>
            <paridad>0</paridad>
            <bitsParo>1</bitsParo>
        </DispositivoSerieMultihilo>
</GeneradorEventos>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Posicionamiento>
    <nombre>posicionamiento</nombre>
    <numeroDatos>5</numeroDatos>
    <datos>
        <Fecha>
            <nombre>fecha</nombre>
            <fecha>11-04-2002 15:11:16.11411</fecha>
        </Fecha>
        <CoordenadasGeograficas>
            <nombre>posicion</nombre>
            <latitud>0.67</latitud>
            <longitud>-45.3</longitud>
        </CoordenadasGeograficas>
            <Real unidadesFisicas="grados">
          <nombre>rumbo</nombre>
            <valor>45.4</valor>
        </Real>
        <Real unidadesFisicas="m/s">
            <nombre>velocidad</nombre>
            <valor>10.1</valor>
        </Real>
        <Real unidadesFisicas="m">
            <nombre>profundidad</nombre>
            <valor>1123.8</valor>
        </Real>
    </datos>
</Posicionamiento>
```

3.3 Application interoperability and remote operation. CORBA

CORBA (Common Object Request Broker Architecture) [9], [10], as a standard middleware developing application, is the answer to the need of interoperability between the different platforms. With CORBA, users can access information no matter what hardware or software platform it resides. A high performance networking and other additional developments in the field of IT as Java and Jini technologies, allow distributed computing to be implemented using a relatively simple and friendly environment. Java Virtual Machine abstracts applications from operating system, allowing the definition of clients' (or server) applications anywhere.

The CORBA architecture is built of four main elements: 1) Object Request Broker (ORB) defines the CORBA object bus, 2) CORBA_*services* define the system-level object frameworks that extends the bus, 3) CORBA_*facilities* define horizontal and vertical application frameworks that are used directly by business objects, and 4) *Aplication_Objects* are the business objects and applications. They are the ultimate consumers of the CORBA infrastructure.

The Interface Definition Language (IDL) is a purely declarative language, which separates the definition of an object from its implementation. With IDL you give clients a way to access and invoke your distributed objects. An interface specifies the contract between code using the object and the code implementing the object. Clients only depend on the interface. Clients can only invoke methods on your objects that are defined in de IDL file. In the IDL file, you provide interface declarations and method signatures. To provide implementations for the interfaces declared with IDL, a mapping between IDL and the language you choose to use has to exist. By using the Object Management Group IDL, the following can be described without regards to any particular programming language:

- Modular object interfaces
- Operations and attributes that an object supports
- Exceptions raised by an operation
- Data types of an operation return value, its parameters, and an objects' attributes

The mappings from IDL and Java are:

- General constructs
- Primitive types
- Constructed types

The Object Request Broker (ORB) is the distributed service that implements the request to the remote object. It locates the remote object on the network, communicates the request to the object, waits for the results and, when available, communicates those results back to the client. The ORB implements location transparency. The client and the CORBA object regardless of where the object is located use exactly the same request mechanism. It might be in the same process with the client, or on a completely different one. The client cannot tell the difference.

At present we have developed a client/server application in Java (Java 2 Runtime Environment) using CORBA and Visibroker ORB for Java 4.5. Programs interact in order to acquire and process raw data. Raw data is not transmitted, instead, only processed (useful scientific data) and statistics are transmitted to remote client. The server program is designed to be executed in several machines at the same time, doing different experiments in each one. The program discriminates which request correspond to which experiment and creates the object implementations serving CORBA functions to other applications. These functions are called by the client programs.

The following box show an example of Java function creating CORBA objects.

SERVER.JAVA

```
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
org.omg.CORBA.Policy[] policies = {
      rootPOA.create_lifespan_policy(LifespanPolicyValue.PERSISTENT)
    };
POA myPOA = rootPOA.create_POA( "sande_agent_" + nomserver, rootPOA.the_POAManager(),
                                policies );
myPOA.activate_object_with_id(managerId, managerServant);
rootPOA.the_POAManager().activate();
orb.run();
```

CLIENT.JAVA

```
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
byte[] managerId = "SandeManager".getBytes();
ManagerHelper.bind(orb, "/sande_agent_" + nomserver, managerId);
```

4. CONCLUSIONS

Using UML we have modeled a distributed acquisition system to be implemented in the Spanish oceanographic vessels. We have identified three different entities (data, logical elements and physical elements).

User interfaces based on Web technology, will allow friendly accessibility from any place with HTTP connectivity. Experiments tools will be accessible using any kind of the available Internet browser.

A brief description of involved technologies is given, 1) Java in order to have code portability and multiplatform possibilities, 2) XML in order to get data portability, and 3) CORBA to achieve interoperability and remote operation.

Mobile agents' technology, using Java RMI (Remote Method Invocation), could be the answer to the development of remote controlled systems. Mobile agents will be used as the transmission agencies of measurement information and must be regarded as computing entities in a collaborative computing environment. The agent will be downloaded from a server inside the experiment platform. The mobile agents' technology will allow the application to move through the network and to be executed from any place, doing their job acquiring data in different places. The use of a communication protocol based on TCP/IP extends the ORB to remote platforms in a way that they can be controlled and executed remotely without limitations.

5. REFERENCES

[1] R. Boza. "Sistema de Adquisición de Datos Oceanográficos (SADO)", UGBO-CSIC-Inf., March 2000.

[2] C. Samitier, E. Trullols, A. Manuel, J. del Río, "Virtual Distributed Instrumentation in the Substation Environment"., Cigre, Paris, France. 2002.

[3] E. Trullols, C. Samitier, A. Barba, R. Morillas. "Entorno de gestión distribuida para la monitorización y correlación de datos oceanográficos", Jitel'01, Barcelona, September 2001.

[4] E. Trullols, A. Manuel, C. Samitier, R. Morillas, "A New Platform for the Observation and Processing of Oceanographic Data", SAAEI'01, Matanzas, Cuba, September 2001.

[5] Boch G., Rumbaugh J., Jacobson I., "The Unified Modeling Language User Guide, Reading, MA", Addison-Wesley, 1998.

[6] Douglass B. P., Doing Hard Time: "Developing Real-Time Systems with UML, Objects, FrameWorks and Patterns", Addison-Wesley, 1999.

[7] Douglass, B.P. Real-Time UML: "Efficient Objects for Embedded Systems", Addison-Wesley, 1999.

[8] Working Group on Marine Data Management. Ottawa, Canada, May 1999.

[9] OMG "The Common Object Request Broker: Architecture and Specification" Technical Report, Revision 2.2, February 1998.

[10] OMG "CORBA Services; Common Object Services Specification" , November 1997.

Acknowledgements.

LabVir project is supported by the Spanish Ministerio de Ciencia y Tecnología under contract TIC2000-1027.