Tests on prototype code for a Distributed Control System Based on JAVA and CORBA

G. Bassato, S. Canella

INFN, Laboratori Nazionali di Legnaro

I. INTRODUCTION

SPES (Study for the Production of Exotic Species)[1] is a L.N.L. project for a facility to produce Radioactive Ion Beams (RIBs) as yield of fission fragments produced by secondary neutrons. It will be characterized by moderate size, performance and cost.

In the context of this design study and tightly related to the medium size of this new facility, the architecture of a distributed control system using the Common Object Request Broker Architecture (CORBA)[2] as middleware framework and JAVA[3] as main programming language was investigated for the core components (diagnostics, optics, RF-control) of the primary accelerator.

In fact, some of the design guidelines for the software of the control system for future L.N.L. accelerators will be portability and openness: that is applications must be portable on different hardware/software platforms (technologies) and will must be able to exchange data among them, no matter if different hardware architectures and/or operating systems will be used. These goals may be easily met using JAVA, an environment which is available also on low cost PCs with Windows or Linux. A good level of modularity and long-term maintenance were some further reasons to choose JAVA and its related Integrated Development Environments (IDEs) as the main programming language and software platform of the new systems.

The performances of CORBA middleware for the high level control system were measured by means of prototype code in different conditions and showed to be sufficient to cover the requirements for remote operations (all feedback loops will be performed either in specialized hardware or by dedicated real-time embedded controllers).

The results of these tests will be here given and discussed.

II. THE CONTROL SYSTEM OF SPES PRIMARY ACCELERATOR

The core components (beam diagnostics, beam optics, RF-control) of the primary accelerator will be controlled by a distributed system, that is by a set of heterogeneous computers, modular systems, embedded controllers and field devices whose objects (data, commands, trends, alarms ...) have to be sharable among local operators and remote users.

From the hardware (HW) point of view, a standard three-level architecture is foreseen for the control system:

- PCs and/or Workstations (WS), at the top level, will be used through graphic interfaces for control and supervision by operators and users, for data storage and general services (network routing and firewall, online documentation, data storage and printing);
- at the middle level there will be VME based systems with PowerPC processor boards and any other specialized board suitable to manage IN/OUT analog and digital signals directly connected to the field;
- embedded controllers and field devices at the bottom level will perform local self-contained tight interactive activities and fast feedbacks.

The connection between top and middle level is planned through a standard fast ethernet (switched 100BaseT), while different heterogeneous connections using different protocols (thin wire ethernet, fieldbuses, serial lines) might be used between middle level systems and field devices or embedded controllers.

Different operating systems (Solaris, Linux, Windows) will be equivalent environments for the top level systems, while VxWorks will be used as the main real-time operating system for the middle level layer and for embedded controllers.

Java applications and applets on PCs and WSs (Java platforms) will establish the SW framework for the Graphical User Interface (GUI) and general services. C or C++ will be the basic programming languages for applications for real-time VME systems and for embedded controllers.

Communications on the network might be based on CORBA middleware. Low level transport mechanisms such as BSD sockets will be also used, if necessary.

Nowadays many IDEs are available for software developments on Java Platforms, especially to produce sophisticated GUIs. Among them, JBuilder from Borland was chosen and put under test because it includes ready beans to connect GUI applications to the Visibroker ORB which is included in the JBuilder Enterprise Edition and works both on top level systems environments (Solaris, Linux, Windows) and on real-time midDle level systems (Vxworks).

III. TESTS PERFORMED

In order to test CORBA middleware and Java IDL some Java tasks were implemented, among them there is a client-server couple with the capability to measure the elapsed time of a typical control transaction: the trip-time of a message between CORBA-based Java applications, eventually on the network (the trip being the transit of a message from the client to a server and back). In other words the subject of these first tests was the typical delay that has to be expected for a service request in a CORBA-based distributed control system. This applications include an IDL module describing the available interface, a
transient server (a program that contains the implementations of the IDL interface), and the code of a CORBA client invoking the available operations on distributed objects. As naming service the `tnameserv` utility (from SUN) was used. The length of the message was changed in a wide range of meaningful values. This test set of applications was used in four different HW-SW configurations:

- Test n. 2: all tasks (nameserver, server and client) running on the same Sun-Solaris8 ULTRA10 WS;
- Test n. 3: server and nameserver running on a Sun-Solaris8 ULTRA10 WS, client running on another Sun-Solaris8 ULTRA10 WS, the two WS connected by a 10 Mb/s ethernet link with no other traffic;
- Test n. 4: server, nameserver and client running on a PC-linux (with an old 200MHz processor);
- Test n. 5: server and nameserver running on a PC-linux (200MHz processor), client running on Sun-Solaris8 ULTRA10 WS, the two computers connected by a 10 Mb/s ethernet link with no other traffic.

All tests were performed by a client sending a message of fixed length and receiving its echo from the server: the trip-time was taken by the client as the elapsed time between the `send` operation and the `receive` operation. Each test was repeated 500 times to have a reasonable statistics. The results (mean, minimum and maximum elapsed times in ms) are summarized in Table 1.

### IV. SUMMARY AND FUTURE DEVELOPMENTS

The performances of CORBA middleware in the above described tests proved to be sufficient to cover the requirements for remote operations: a typical message of 1 Kbyte (a coded request with a reasonable number of parameters) will travel forward from the client to a server and back (with the answer to the request of service) in an average time not greater than 10 ms, in all the four configurations. A greater delay may be noticed from time to time due to the concurrent activities of the systems, but always above the acceptable limit of 100 ms.

From the above described basic choices (CORBA, Java, IDEs) a number of benefits are expected for the control system of SPES primary accelerator that was here described:

- a long lifetime (over 10 years) for the system may be foreseen, both from the HW and SW point of view;
- HW components at the top and middle levels will be available from a wide range of manufacturers;
- high SW reliability is expected, because compile-time and run-time extensive checking will help to have a fast and extensive code debugging;
- Java automatic memory garbage collection will also help to produce reliable code;
- minimum time to repair (HW) and/or for maintenance (HW/SW)
- uniform code (and bytecode) for different Java platforms greatly simplify SW updating and maintenance.

In the next future more tests will be performed, especially using JBuilder as main tool of SW design and implementation:

![Table 1 - Tests results of CORBA-JAVA applications to measure the trip times for messages of different lengths](image)