A Web Services based system for the distribution of live information at the FTU fusion experiment

L. Boncagni a,*, C. Centioli b, L. Lattanzio b, M. Panella a, C. Torelli a, L. Zaccarian b

a Associazione Euratom/ENEA sulla fusione, Centro Ricerche Frascati, CP 65 - 00044 Frascati, Roma, Italy
b Dip. di Informatica, Sistemi e Produzione, Universitá di Roma, Tor Vergata, Via del Politecnico 1 - 00133, Roma, Italy

1. Introduction

As in most long lasting experiments, the Frascati Tokamak Upgrade (FTU) control and data management systems have undergone several upgrades during their lifetime. So far, the FTU Broadcast system – a system devoted to the display of live information in the Frascati research center – remained roughly the same over the years. Recently, an increased demand for a more flexible system has led to the development of LiveMonitor: a new information distribution system that will replace the current one, introducing furthermore the possibility of upgrades. The system relies on a client–server architecture, where the server side consists of a set of Web Services that collect data from a variety of data sources. LiveMonitor has been successfully used at FTU, replacing and enhancing part of the core of the current message broadcasting system. The tool integrates all the information needed by the control room personnel during the experiments, namely the shot sequence status coming from the FTU Control System, videos of the plasma discharge from the FTU ports cameras, and fresh data from the databases. From the hardware point of view, the new system is made of a Linux node running the Web Services, while clients running on other machines can display information on large LCD monitors. The tool has been tested during FTU experiments and can be further expanded to match the needs of the control room personnel and experimental physicists.

2. Environment description

The current FTU broadcast system is made of a Macintosh powerPC, a VGA/PAL converter and some PAL TV/Monitors sets located in the FTU control room as well as in other buildings, where people may require to know the live status of the experiment. During an experimental session, the Macintosh acts as a TCP/IP server waiting for status messages from the Control System. Depending on the message content, the broadcast program behavior follows the state diagram shown in Fig. 1 and consequently the experiment evolution, then dispatching the results of its own elaboration to the TV sets using the VGA/PAL converter.

The events characterizing the typical shot evolution can be roughly summarized as follows: in the initial state (Sequence Start in Fig. 1) the FTU data acquisition system is initialized. In the Pre-Run phase, lasting 120 s, the FTU Control System initializes all the subplants configured for the specific experiment. The Start Run state triggers the so called fast (i.e. fully hardware controlled) phase of the experiment, lasting 30 s and ending with the End Run phase, when the acquired data is collected and archived and the FTU subplants are brought back to their idle state.

The layout of the broadcast system varies according to the sequence evolution, with the exception of some persistent information such as the name of the physicist in charge, the scientific program name, the discharge name and the current date. In the initial state the broadcast system shows the shot number, the status message and the other persistent information. During the Pre-Run
3. Architecture

The new version of the broadcast system fixes the above mentioned problems using a client–server concept, aiming at enhancing its modularity, flexibility and scalability. In fact, the new architecture relies on a server which can access to a potentially unlimited variety of data sources. Clients can select autonomously the data needed among the information provided by the Web Services on demand, and they will deal with the issues related to data display.

Based on these requirements, to realize the server side of the project Web Service technology has proven to be the most convenient. A Web service is defined by the W3C [1–3] as being “a software system designed to support interoperable machine-to-machine interactions over a network”.

Normally Web Services are Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services. The main advantages in using this technology rely on:

- Little effort in development, thanks to automatic code generation.
- Clients can be implemented in all those languages for which a SOAP/XML library exists.
- SOAP is firewall compliant, using HTTP as transport protocol.

The server is fully developed in Java using Apache Tomcat plus the Axis2 module as Web Services container, as shown in Fig. 2. The Web Service objects are instantiated by the Axis2 module each time a request coming from a client must be processed.

For the whole broadcast system a set of three Web Services is provided, actually one for each data format:

- shot sequence status message;
- videos of plasma discharge;
- data from databases.

Each Web Service publishes at least two methods: getCurrentData() and getNextData(). The first one is a nonblocking call that returns the current data to the requesting client, the second one because of the small rate of status change (as described in Section 2) is a blocking call that returns the fresh requested data as soon as it is available. In any case, due to the Web Services client APIs capabilities, the methods can be invoked in synchronous or asynchronous mode.

Each Web Service instance is linked to one or more static receiver threads (static: the same threads for each Web Service instance) that deals with its own data source. In case of a getNextData() invocation, the corresponding Web Service instance is queued if the receiver is busy (i.e. waiting for fresh data). The concurrency between the receiver and the Web Service instance is handled using locks on shared objects and some synchronized methods or code sections. Only when the receiver understands that the data is ready for delivery, the locks on the shared objects are removed and consequently notified to all the Web Service instances that are queued waiting for the resource. After the delivery, the receiver will take again the locks and the whole system returns to its initial state. A schematic view of the Web Services definition is shown in Fig. 3.

In the next three subsections we will describe briefly the three above mentioned services.

3.1. Message WS

The Message Web Service is in charge of one way message delivery from the FTU Control System. The messages are intercepted by the message receiver thread. As in the current system, it acts like a server waiting for TCP/IP communication, but during the transmission of the message, a copy is stored into a database table for...
a configurable time. The TCP message status is a simple pair \((id, value)\) that represents the state \(id\) and an additional info \(value\). For example, the pair \((1, 30478)\) communicates the Sequence Start of shot 30478.

3.2. Database WS

The database Web Service deals with the information stored in databases. The data source of this service is the FTU Logbook [4], an application that manages relevant data about experiments entered by both physicists in charge and the Control System, and are stored via a MySQL DB engine.

Data is made available creating a view as a select from the Logbook data table, with some temporal and order constraints. The view content is limited to the current date and the last five shot events, so that a simple selection represents a sliding time window of some fields on the Logbook data. In this case the receiver thread waits for changes in the data using an hash function on them. Since a view is used, it is also possible to redefine it for a new selection without restarting the system. The Database WS also provides a method to retrieve the last complete set of persistent data, stored by the Message WS.

3.3. Video WS

During the plasma discharge, three CCD cameras placed inside three different ports and controlled by three LabVIEW applications [5] acquire with a configurable rate (maximum 50 frame per s) three set of jpg frames. The LabVIEW applications are responsible for:

- storage of the jpg frames;
- creation AVI movies (one for port);
- sending the acquired frames to the instanced receiver threads.

The client can ask for a contiguous set of frames of the next video \(\text{GetNextVideo()}\), configurable on the client’s side. The Video WS will deliver the required frames as a unique base 64 encoded string. The WS provides the \(\text{getLoadValue()}\) and \(\text{getVideo()}\) methods too, the first one to know the percentage of frames (arrived/requested) unlocked by the receiver thread, the second one to retrieve frames related to past events.

3.4. Client side

The client side of the architecture consists in a client that reproduces exactly the behavior of the current system as shown in Fig. 4, except for the fact that the video stream of the event is integrated into the shot sequence (anyways, it is still reproduced in real time by the PAL TVs that are directly connected to the cameras). The client can be configured to display one, two or three port cameras. The client side has been implemented in two versions, the first one in Java Language (Fig. 4) as an application, and the second using a Flash applet for an easier browser integration (Fig. 5).

3.5. Hardware

From the hardware point of view, the new broadcast system is composed of three new flat LCD HDI Ready 46” monitors, each one of them plugged into a standard PC, located in the control room. Although the software does not have any special requirement, the PCs should be equipped with a Pentium 4 or better processor, and
Fig. 4. Experiment Sequence Java Broadcast Reproduction.

Fig. 5. Experiment Sequence Flash Broadcast Reproduction.
should have at least 1 Gb of RAM. The operating system is Ubuntu Linux. One of the three PCs runs both the client and the server side of the architecture, while on the other ones only the client has been configured, depending on the needs, with different ports cameras.

4. Testing and conclusions

The new system was developed using a test program that sends messages and videos to simulate the behavior depicted in Fig. 1: the screenshots shown are made using this test tool. Thereafter, the Control System and the LabVIEW application were configured to send information to the new broadcasting system too, so thanks to the FTU test facility it has been possible to make tests on the real environment without waiting for a dedicated experimental session. Up to now a test on a set of only 22 shot events was possible. During this test the current and the new system were running simultaneously, so that the control room personnel could check upon the correct reproduction of the events.

The new architecture opens brand new perspectives in the FTU information sharing: in fact, thanks to its modularity and scalability, other clients (i.e. desktop applications for browsing images and videos, or programs that alert on the state of the experiment) could be developed using the Web Services interface. Finally new services can be added to the Web Service container to implement additional required facilities.

Acknowledgements

This work was supported by the Euratom Communities under the contract of Association between EURATOM/ENEA. The views and options expressed herein do not necessarily reflect those of the European Commission. The authors wish to thank Dr. Riccardo De Angelis and Enzo Vitale for their invaluable help and advice.

References