Fusion Engineering and Design 84 (2009) 475–479

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

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

L. Boncagni[a,*], C. Centioli[a], 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, Università di Roma, Tor Vergata, Via del Politecnico 1 - 00133, Roma, Italy

ARTICLE INFO

Article history:
Received 7 August 2008
Received in revised form 9 December 2008
Accepted 9 December 2008
Available online 29 January 2009

Keywords:
FTU
Broadcast
Web Services

ABSTRACT

In this paper we describe LiveMonitor, an integrated system realized for the distribution of information in fusion environments. The software tool is based on a client–server approach, 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 (46″) 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.

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 clients can pick up the requested data from the information provided by Web Services on demand. Data coherence among the different clients is guaranteed by a refresh mechanism on the server side: one or more receiver threads for each Web Service waits for data from a selected source; when the data is ready the receivers collects it and all the Web Service instances generated by clients requests wake up and deliver a copy of the information to the clients. In the following sections a short description of the system will be given, together with implementation details and hints on further developments.

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

* Corresponding author. Tel.: +39 06 9400 5245.
E-mail address: boncagni@frascati.enea.it (L. Boncagni).

0920-3796/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.fusengdes.2008.12.055

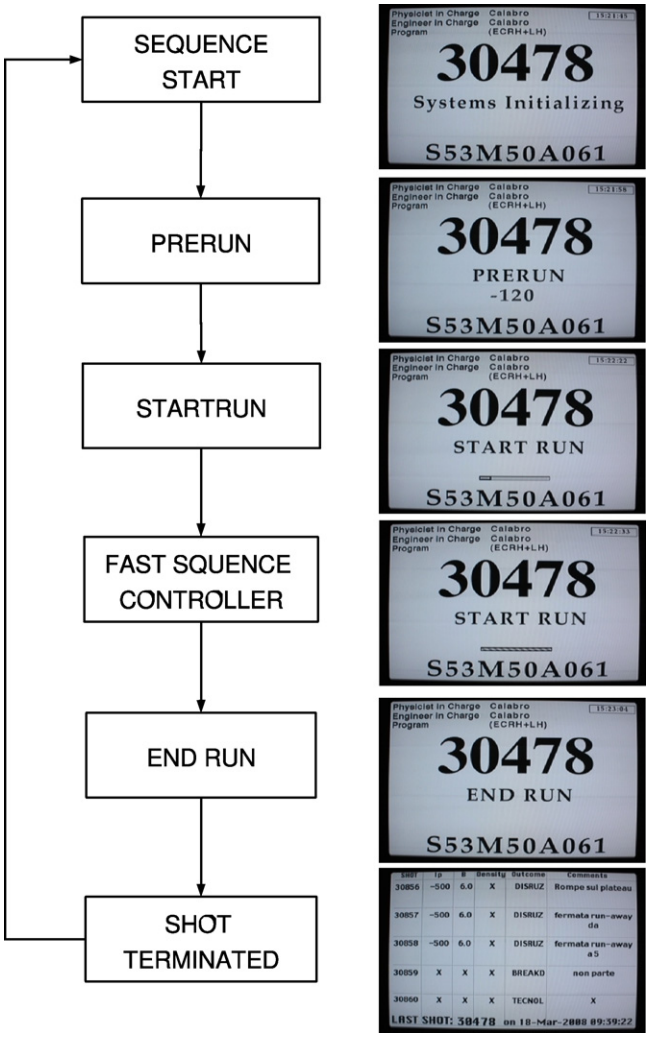

Fig. 1. Broadcast Sequence and Current System.

state the broadcast screen shows a countdown status bar, which changes color and appearance when the Start Run and the Fast Sequence Controller (FSC) states are reached. The End Run state has a variable duration depending on the amount of data that the Control System is configured to elaborate and then transfer to the archive. The broadcast represents this state likewise the Start Sequence state.

In the idle state between two subsequent experiments (Shot Terminated in Fig. 1) all the data is ready and the broadcast summarizes in a table important measurements and parameters (shot number, shot result, plasma density) relevant to the last five experiments.

This system presents several disadvantages. The first – and most obvious – is that the TV sets are located in 'fixed' places, connected through dedicated optical fiber and coaxial cables. Another disadvantage is that the system preserves no memory of the broadcasted live data. This is particularly annoying especially in the case of a broadcast system restart, when the persistent variables normally displayed on the TV screens (i.e. the scientific program name, the name of the physicist in charge) are considered as unknown until they are transmitted again. Finally, due to the intrinsic lack of modularity of the current broadcast system, it is quite difficult to add or modify any software functionality of the system. A typical example is when a new data type is required to be displayed, and the whole broadcast data structure must be modified to handle it.

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