Informes Técnicos Ciemat **1098**
Febrero, 2007

Visual Data Analysis in the TJ-II Remote Participation System

E. Sánchez
A. Portas
A. Pereira
J. Vega

Laboratorio Nacional de Fusión por Confinamiento Magnético

Visual Data Analysis in the TJ-II Remote Participation System

Sánchez, E.; Portas, A.; Pereira, A.; Vega, J.

20 pp. 4 figs. 16 refs.

Abstract:
A general-purpose data visualization tool has been developed to provide the TJ-II remote participation system with the same visualization capabilities already available in the TJ-II local environment. The visualization software has been developed in the Java language. It provides a user-friendly graphical interface that permits users on-demand plotting of time traces in a very flexible manner. In order to facilitate on-line tracking of experimental operation, the application also allows automatic refreshing of data. This software has been integrated into the TJ-II remote participation system distributed environment. Data are accessed remotely using web technologies and HTTP protocol and are transferred in a compressed format, which reduces bandwidth requirements. Both metadata and binary compressed data are transported in multipart messages. Message oriented middleware software is used to distribute information on-line, in particular notifications of data availability for automatic data refreshing or local events. Plot layouts can be stored in a centralized database for subsequent recovery from anywhere. Finally, this software is integrated into the general security framework provided by the PAPI system.

Análisis Visual de Datos en el Sistema de Participación Remota de TJ-II

Sánchez, E.; Portas, A.; Pereira, A.; Vega, J.

20 pp. 4 figs. 16 refs.

Resumen:
Se ha desarrollado una herramienta de visualización de datos de propósito general para dotar al sistema de participación remota de TJ-II de las capacidades de visualización de datos ya disponibles en el entorno local. El software de visualización se ha desarrollado en lenguaje Java y proporciona una interfaz de usuario amigable que permite a los usuarios visualizar bajo demanda trazas temporales de una manera muy flexible. Para facilitar el seguimiento en directo de la operación, la aplicación proporciona también la funcionalidad de refresco automático. Este software se ha integrado en el sistema de participación remota de TJ-II. El acceso a los datos se realiza usando tecnologías web y el protocolo http y los datos se transfieren en un formato comprimido para reducir las necesidades de ancho de banda en la red. Los datos binarios comprimidos se transportan junto con meta-datos en mensajes multiparte. Para distribuir información en directo se usa un software intermedio orientado a mensajes. En particular, se distribuyen notificaciones acerca de la disponibilidad de datos experimentales para el refresco automático o eventos locales. El diseño de los gráficos se puede almacenar en una base de datos centralizada para ser recuperados con posterioridad desde cualquier parte. Finalmente, este software se integra dentro del entorno de seguridad proporcionado por PAPI.

CLASIFICACIÓN DOE Y DESCRIPTORES

S70
TOKAMAK DEVICES; DATA ANALYSIS: REMOTE SENSING; COMPUTER OUTPUT DEVICES; JAVA; STELLARATORS; CALIBRATION

INDEX

1. Introduction

TJ-II is a medium size stellarator (R = 1.5 m, a = 0.22 m, B ≤ 1.2 T) operated in the Laboratorio Nacional de Fusion, Madrid, Spain since 1997 [1]. The Data Acquisition System (DAQ) developed for TJ-II [2] has provided users with software tools for data storage, data acquisition control and programming, as well as for basic data visualization within its local area network. The acquisition of data was originally carried out with VXI and CAMAC systems [3,4]. During machine operation all acquisition processes were commanded from a central ALPAH/AXP Tru64 UNIX server where the acquired data were also stored in a multilayer database MLDB [5]. In the original system the users programmed their desired parameters for the acquisition systems through a Graphic User Interface (GUI). The DAQ also provides users with a software tool for fast visualization of data during operation [2]. This visualization tool provides, among others, the useful feature of automatically refreshing acquired data when a new plasma discharge takes place in TJ-II. It has been routinely used in the TJ-II control room to follow operation since the stellarator start-up phase in 1997. Almost the same data visualization functionalities were provided by a second application that was developed for other computers other than the central server, thereby reducing the consumption of server computation resources by the visualization tools [6]. However, this application does not provide an automatic refresh feature.

In recent years the TJ-II DAQ has received some major upgrades: i. e. new acquisition systems, based on PCI and PXI standards, were incorporated to the TJ-II DAQ [7] and the TJ-II Remote Participation System (RPS) was designed and developed [8,9]. The design of the RPS focused on providing tools to follow discharge production, reading/writing information in the databases, and performing simple visual data analysis

from outside the laboratory. In a first step, tools for remote data acquisition programming, remote diagnostic control and remote access to the logbook were developed.

In the present paper a recently developed general-purpose data visualization tool is presented. It is designed to provide the TJ-II RPS with the same visualization capabilities already available in the TJ-II local environment. The motivation, the architecture and the characteristics of this software are described.

2. Motivation

The original visualization software included in the TJ-II DAQ [2,6] was developed in the C programming language. It made use of the X Toolkit (Xt) and Motif (Xm) libraries, that made this software platform dependent. In fact the visualization application is only available for an ALPHA/UNIX platform. Two versions of the visualization application are available. The first version is devoted to "on-line" visualization of data; it runs on the central server and it is synchronized with the rest of acquisition tasks running in the server by means of local inter-process communication (IPC) tools: shared memory, UNIX signals, FIFOS and pipes [2]. Use is made of these tools to provide automatic refreshing of user-selected data when a new plasma discharge takes place in the TJ-II device. Rapid access to the most recently acquired data is possible because of the use of a large shared memory area in the central server. A second visualization application was provided for visualizing data without synchronization with central server DAQ processes. This application can be run in any ALPHA AXP/UNIX computer. It uses the Remote Procedure Call (RPC) data access routines [10] for reading data from the central database.

These visualization tools have been used extensively for many years and have proved to be very flexible. However they present some drawbacks for current situation. First

of all, they can only be run in the ALPHA/UNIX platform; the porting of these applications to other platforms would require a significant programming effort. Generally, GUI applications require a lot of code lines that usually are very platform dependent; in the present case, the GUI application for data visualization has about 30,000 of C code lines and hundreds of calls to functions from Xt and Xm graphic libraries. Secondly, these applications were designed to work in a local environment only. The on-line version runs on the central computer and requires local IPC tools for synchronization with TJ-II operation. The "off-line" version, although it does not use local IPC's and can access data remotely, uses Open Network Computing (ONC) RPC protocol to retrieve data that presents some problems for use in Wide Area Networks (WAN). First, it requires an explicit access authorization to ONC RPC ports in the institutional firewalls that sometimes is not accepted because it is considered a non-secure protocol. On the other hand data are transported from the central server to the client application in the machine independent but non-compressed format External Data Representation (XDR) that implies a payload for translation. Indeed this is not a good option for very large data transfers.

Finally, Personal Computers (PC) running MS Windows operating system have substituted the old ALPHA AXP/UNIX workstations in the TJ-II control room. As a result of this the possibility to use the off-line version of the application even within the local area network is reduced.

Taking into account the limitations in current software it has become inevitable to design new visualization software to provide remote users with almost the same basic tools for data visualization and analysis that are available in the local environment. Now, in order to maximize exploitation of the development effort, the newly developed tools should be available also for the local environment, thus providing an upgrade to present tools.

3. Software architecture

The software has been designed following the distributed architecture of the TJ-II RPS into which it is integrated. We can distinguish two main parts of the software, firstly the Java Application for Signal Visualization (JASVI) that acts as a client application which can be run in any computer in the network. This application is described in some detail in the next section. Second a software infrastructure is installed locally to provide this application access to data and resources for synchronization with the local environment.

The client visualization application is intended for use not only in the TJ-II local environment, but also in remote computers connected to the Internet. Taking this into account, it should be as multiplatform as possible. It has to be usable in Wide Area Networks (WAN), so that the data transfer from the local environment to the client application should be optimized for this end. It should also be taken into account that most institutions have firewalls to preserve security. The software should be as transparent as possible to the presence of institutional firewalls. Considering the environment for which this software is intended, software deployment should also be as transparent as possible. Finally, because the software can be used in many remote clients, security is an issue. Access to local resources should be protected. Most of these requirements are the same as those imposed for the TJ-II RPS from the design phase. Bearing in mind the remote participation environment for which this software is developed it is worth to provide the functionality of visualizing data from several different fusion devices simultaneously

3.1. Programming languages

To address the multiplatform requirement the visualization application has been developed in the Java programming language, which was also chosen as the language for TJ-II RPS applications. An important drawback of using Java is that the application

performance can decrease the byte code being interpreted by the Java virtual machine. This drawback can be compensated in part by the continuously increasing calculation power of desktop computers. In contrast, the advantage is that Java provides not only multiplatform support but also an enormous number of libraries of already developed components. In particular there are a lot of components that provide support for a good integration in distributed environments using the standard protocols, Transmission Control Protocol (TCP) and HyperText Transfer Protocol (HTTP) among others.

Some Java Server Pages (JSP) have been developed to provided access to the TJ-II experimental data. These pages are served by Apache/Tomcat.

In addition, the C language has been used for developing a data server (DS) running in the central database host that is in charge of the integration into, and the recovery of data from, the central database.

3.2. Data access scheme

Access to the experimental data is performed by means of a client/server architecture. In order to optimize network traffic, data are transferred from the server to the client in a compressed format. The compression methods used here are the same lossless methods used in the TJ-II MLDB [11] allowing on average savings of about 70% of the disk space required to store data without compression that now allows us to decrease the bandwidth necessities by the same factor. Data are directly read from the central database in the compressed format; they are sent to the client without decompression, and are decompressed in the client application. This also allows an enormous improvement in the data transfer as compared to the RPC access routines that transferred the data from the server to the client application in XDR format without compression.

A server program (DS) has been developed in C language to read the data from the central TJ-II database. This server uses Berkeley sockets for communications and sends/receives data in a compressed format. Data are accessed from the client visualization application (see next section) through a Java Server Page (JSP) served by Apache/Tomcat. This JSP page executes a client component that calls the data from the central database. The communication between this client JSP page (Java) and the server (C program) is carried out using Berkeley sockets Application program Interface (API). Data are packed into a multipart Multipurpose Internet Mail Extensions (MIME) message and sent forward from the web server to the client application using HTTP protocol. The first part of the message contains the metadata in eXtensible Markup Language (XML) format, describing the acquisition conditions of the data, number of data, and so on. The second (binary) part of the message carries the actual samples in a compressed format. Use is made of the JavaMail API to manage MIME messages both in the JSP and in the client visualization application. A schematic view of the data access architecture is show in Fig. 1.

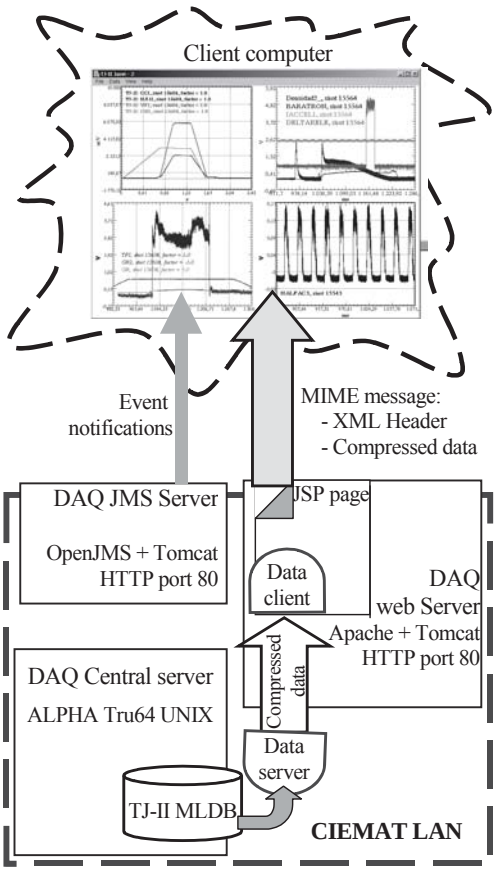

Figure 1. Schematic view of the data access architecture

The use of HTTP protocol for data access and the web server running on port 80 make easier passing through institutional firewalls. Both HTTP protocol and port 80 use to

have allowed access in the firewalls so that it is not necessary to define new firewall rules, neither in the client institution nor in the local one.

3.3. Synchronization with local environment.

Synchronization with local acquisition processes is an important feature that aids users to follow the routine operation of the TJ-II device. Indeed this feature was present in the visualization software available in the local environment. Now for the tools described here the Message Oriented Middleware (MOM) software recently introduced in the TJ-II RPS [12] is used. OpenJMS [13] is used as the messaging broker. Local events are sent to client applications that receive notifications by means of the Java Message Service (JMS) API [14]. Automatic refreshing of data, when a new plasma discharge occurs in TJ-II is implemented in the visualization application by taking advantage of this software. Notifications about the start and the end of a new TJ-II discharge are also distributed to the visualization application by means of the JMS middleware.

3.4. Multi-site access.

The functionality of visualizing data from several different fusion devices simultaneously has been implemented by means of a plug-in architecture. This allows us to maintain a common kernel in the visualization application while dynamically adding different plug-ins that provide specific resources to obtain data and notifications from different devices (data sources).

All the interactions between the visualization application and the TJ-II local environment are implemented through a plug-in that is also loaded from the TJ-II web server. This plug-in provides the tools for calling experimental data, decompressing them and subscribing for receiving event notifications.

The interaction between the visualization application and the plug-in is implemented through two interfaces: JasviDataSource and JasviMainApp. The JasviDataSource interface defines the methods that a data plug-in has to provide in order to be "pluggable" into the JASVI application. The second interface (JasviMainApp) defines the methods in the main application that are exposed and can be called from a data plug-in to interact with the visualization application. A Tj2DataSource class implementing the JasviDataSource interface has been created for the TJ-II connection plug-in. It implements the methods related to the services provided by the plug-in for the main application: to read the list of time signals that are defined in the TJ-II database for which data are acquired routinely, to read the number of the last TJ-II shot and to request the data acquired by a signal in a plasma discharge. The plug-in automatically connects to the TJ-II JMS services in order to subscribe and receive local event notifications (shot start, shot end and data availability). The number of the last shot carried out can be updated synchronously by an interface method that reads it from a JSP page and also asynchronously by the JMS services.

The JasviMainApp interface defines several methods: firstly there are two methods that allow the data plug-in to send the main application notifications about data availability for automatic refreshing and send it previously requested datasets for plotting. In addition, several methods allow the plug-in to query the main application about disk cache settings: availability, cache directory name, and maximum cache size. Finally, another method allows the plug-in to send information messages concerning received notifications or errors occurred in the plug-in code to the main application for display in the log window.

Interaction with other fusion environments can be implemented by means of different plug-ins, while maintaining the same application kernel.

3.5. Software deployment and security

The application described in this paper is downloaded from a web page in the TJ-II web server using Java Network Launching Protocol (JNLP) [15] that facilitates its updating when needed. Also, in this way the user always executes the latest version that is available.

As many remote users can potentially use the visualization software application, it is essential that all accesses are secure. This is warranted by the PAPI system [16] by using also the extensions for Java applications provided by the PAPI development group. Only Authenticated and authorized users can execute the visualization application. Data access is also protected by PAPI and only authenticated users can access TJ-II data.

4. The Java Application for Signal Visualization

The Java application for signal visualization is a software application that allows users to visualize experimental time traces in a user friendly GUI. In addition, it provides both remote and local users with the same, and increased functionalities than were available with the previous local visualization tools. It should be noted that more than 90% of the experimental signals acquired in TJ-II are time traces, this is the reason why the visualization software has focused first on providing tools for visualization of these kind of data.

A minimal visualization application was developed when the MOM software was implanted into the RPS [12] to provide users with a tool for remote visualization of pre-selected time traces. It provided a minimal set of features that have been greatly extended and improved. In this previous application, data were sent to the client visualization software by using the MOM software. This scheme has been abandoned and a new on-demand scheme has been implemented here that is much more flexible and better suited. In the present application, data are always called from the visualization application either as a consequence of a user action or automatically after an event reception (see below).

In the next subsections the main characteristics of the visualization application are described in detail.

4.1. A user friendly Graphic User Interface

The application has been designed to provide users with a tool for very flexible visual analysis of time traces, while allowing them to change most of the visualization settings by simple mouse actions. Data plots can be distributed in a non-limited number of visualization windows, each of which contains a set of visualization areas distributed in rows and columns into the time traces can be plotted. Moreover, several time traces can be visualized in the same area simultaneously. For this the time traces share x and y axis while different offset values and amplification factors can be individually applied to each signal instance. A legend can be displayed also in each area. It provides the user with the data source from which the data came (e.g. TJ-II), the signal name, the amplification factor and the offset applied. In addition, a log window shows information about the result of user-required actions, errors as well as notification of events received asynchronously from different data sources to which the application can be connected.

In order to aid the user in customizing the visualization environment the visualization areas can be copied and pasted into other areas and plot windows can be cloned to facilitate the composition of a user-adapted framework. When copying an area and pasting it into another, all the signal visualization instances from the first area are added to the destination area while maintaining the destination area attributes (i.e., axis, grid, legend, labels, margins, ticks and background color styles).

Next, all the visualization settings selected by a user can be stored thereby allowing the user to recover the visualization status from the previous visualization session, thereby saving on the time and effort necessary to customize a user environment again. These plot

layouts settings can be stored in local files (in the client computer) or in the TJ-II web server for posterior recovery from anywhere. User data stored in the web server are PAPI protected and limited in size. The number of different plot configurations stored in the web server is also limited. The plot settings are stored in XML format thus allowing the portability between different computing platforms. Figure 2. shows the JASVI GUI with a visualization session containing a window with four visualization areas and the log window in the background. A popup menu is also shown. It allows a user changing most of the plot style by mouse actions.

4.2. Visual analysis of data

In order to facilitate the visual data inspection visualization areas can be zoomed in/out with simple mouse clicks and drags. Signals can be vertically and horizontally displaced very quickly by mouse movements and keyboard actions thereby helping in the detailed inspection and comparison of traces. The offset value and the amplification factor applied to a signal visualization instance can be set, or modified, both from a textbox as well as by mouse and cursor movements over the visualization area. The information about the acquisition conditions of a signal (number of bits in the ADC, sampling period, …) can be visualized at any moment.

Finally, smooth filters (Gaussian, Rectangular and Hanning windows) can be applied to signals for noise reduction. These filters are applied only for visualization purposes. The filters can also be recursively applied to a time trace to produce even more smoothed results.

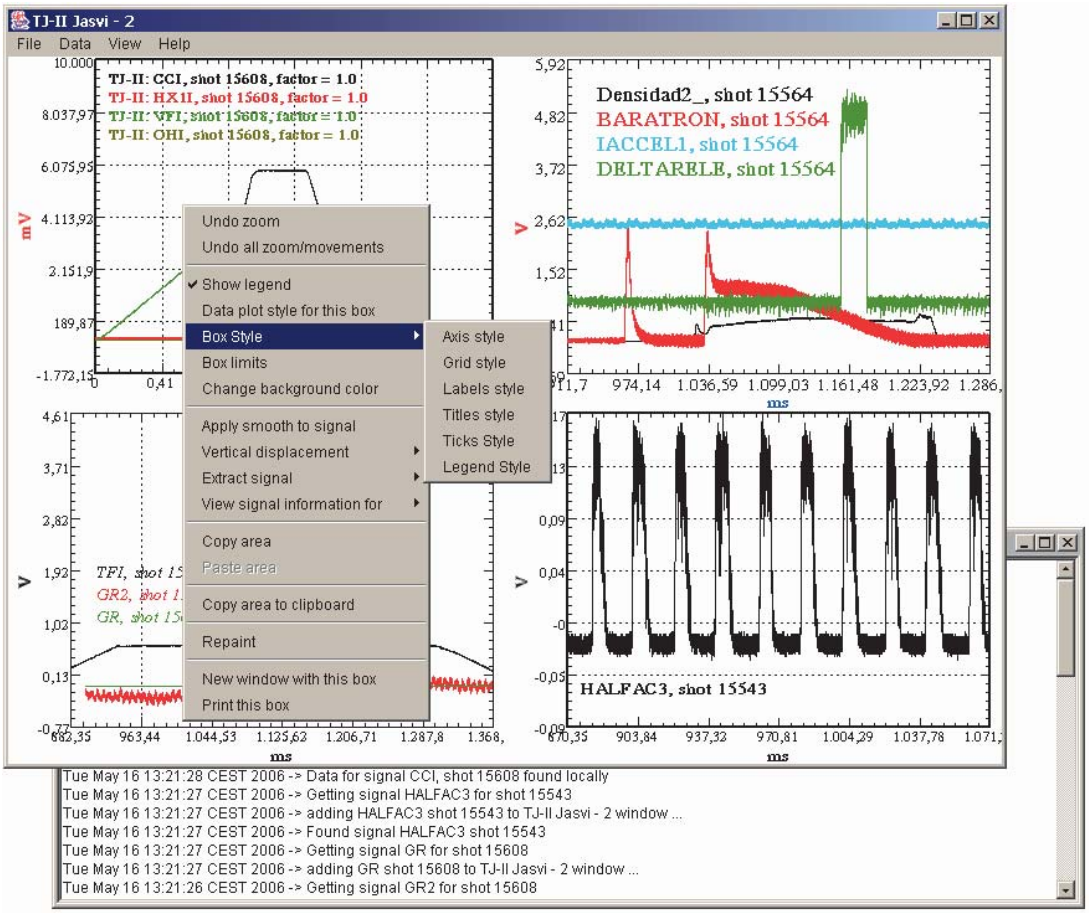

Figure 2. Graphic User Interface of the Java Application for Signal Visualization showing a plot window with four visualization areas and the log window behind. A popup menu is shown with some buttons that allow customizing the plot style.

4.3.

High quality plots.

The aim of this visualization software is to allow a user to perform basic visual inspection and analysis of data while also allowing high quality figures to be produced. The style of the data plot for printing can be customized by the user with mouse clicks. Signals can be plotted with lines in different styles and colors. In addition, symbols can be added to the data points as required. Fig. 3 is an example of a Plot Style window that allows a user to define what signals are plotted in each visualization area and the plot style offset value and factor for them. A user can customize both the content shown and the font for the legends. The user can also show or hide a legend in any area. The axis, tick markers, labels and titles fonts can be changed to produce publication quality figures. A grid can be added to the axis ticks to help in the visualization and analysis of the traces. Plots can then be printed, from a flexible print layout window in which the user can select the visualization areas to be

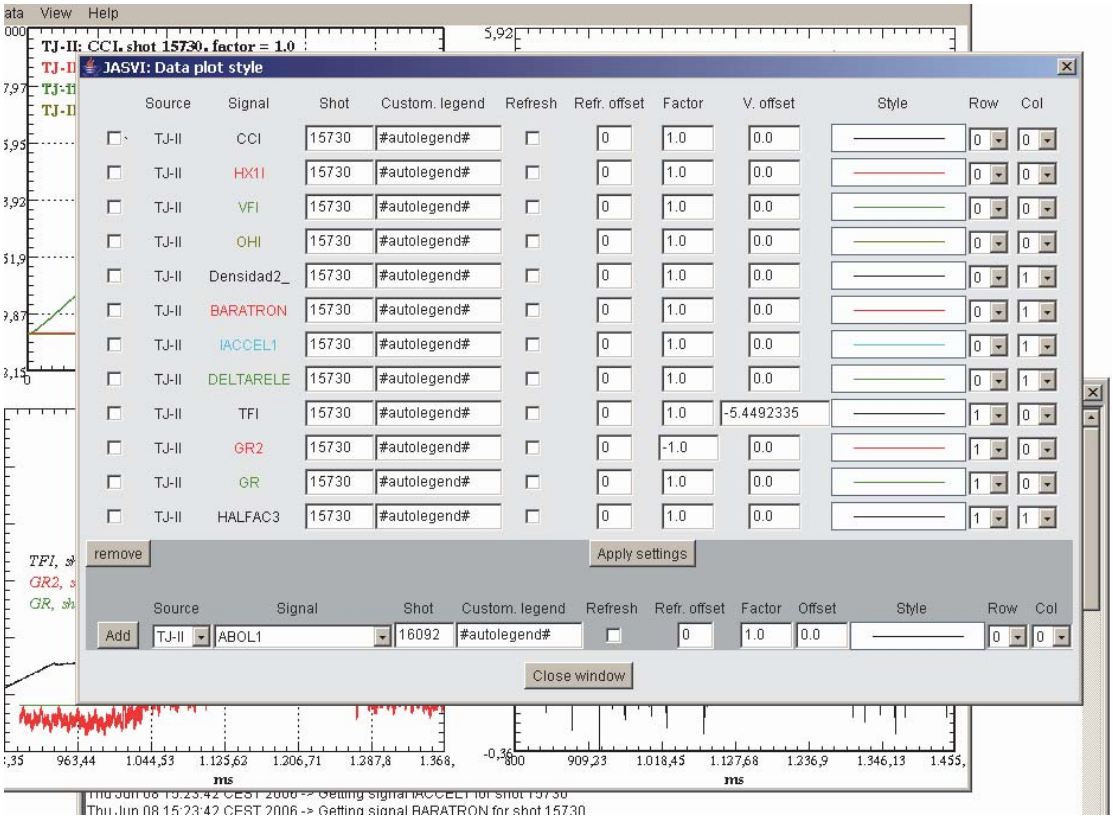

Figure 3. Plot style window

included. Figure 4 shows a Print Layout Window (PLW) where the areas to be printed are chosen. In addition, plots can be copied to the system clipboard and pasted into other applications documents.

4.4. Optimized plots

A time trace can be plotted many times in different (or the same) visualization areas with different amplification factors, offsets and plot styles to allow comparison with different signals. However, for all the visualization instances of the signal only a single copy of the data corresponding to each time trace is maintained in memory in order to optimize memory consumption.

In addition, a special routine has been developed to generate fast X-Y plots. To this end the number of pixels that need to be plotted is minimized. In accordance with the visualization settings selected by the user, the area size and monitor resolution, the extreme vertical positions that a signal reaches for a given horizontal position in the visualization area are computed and only the line joining these two points is plotted. It has been found that this plot routine is much more efficient for plotting experimental traces than using directly the routines provided by the Java package to calculate the pixels that need to be plotted. In the case where a signal has many samples a lot of pixels should be calculated, many of the experimental samples can be mapped into the same pixel; in this case it is better first to calculate the independent pixels and plot only those. When a signal has few points for plotting in the visualization area, it could be better to plot directly, but as there are few samples the time needed for calculation is also very short.

During the visual analysis of experimental data some data may be plotted many times to make cross comparisons. These multiple re-plots of data imply multiple accesses to the same data. To decrease the read time for recently accessed data the visualization application allows using a disk cache, so that accessed data can be stored in the local disk for a faster subsequent access. This feature is very useful when access to experimental data is

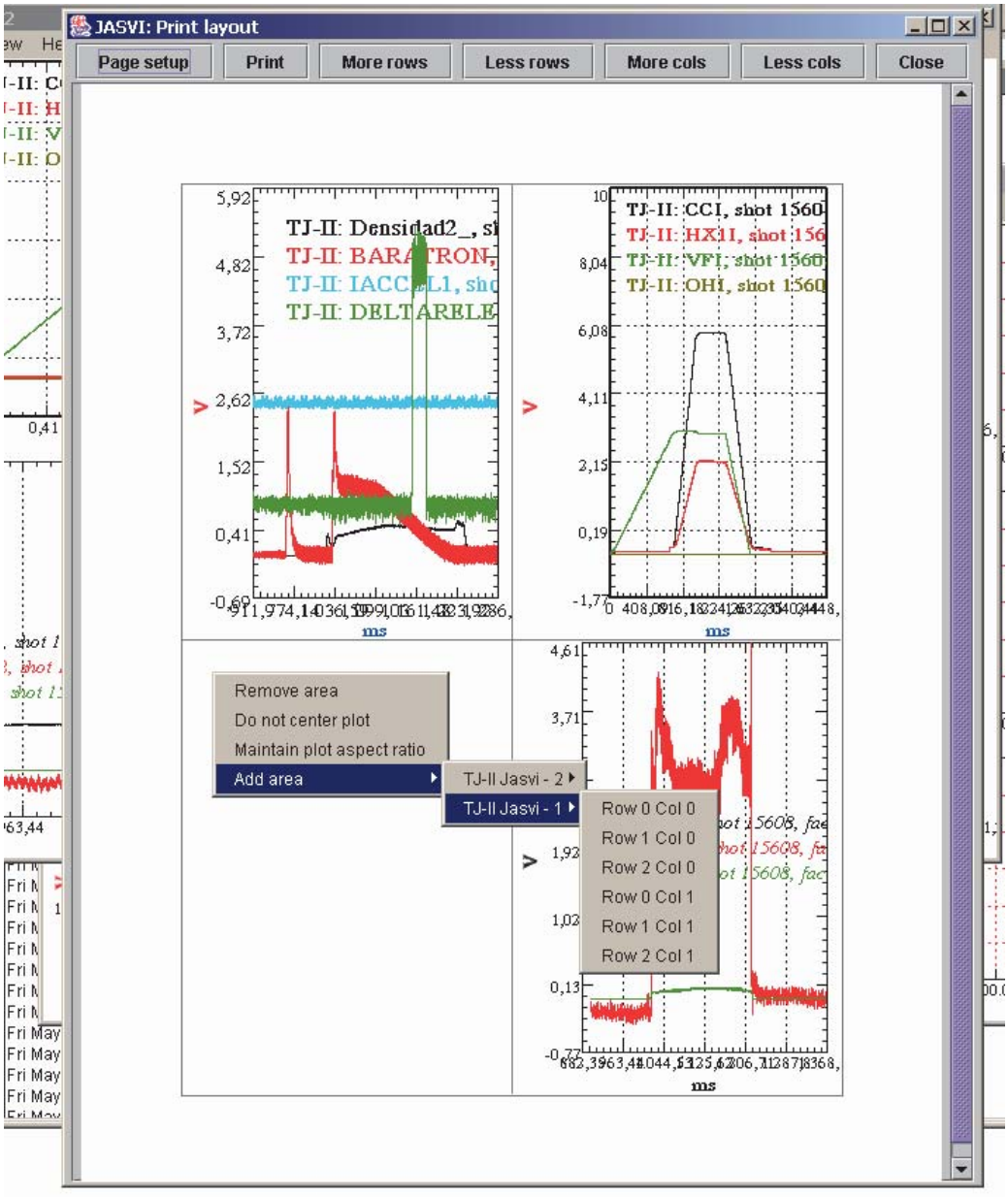

Figure 4. Print Layout window showing the popup menu that allows choosing the visualization areas that are included in the printed page.

done from very disparate sites. It is also useful in the local environment to decrease the data server load by decreasing the number of data calls for recently read data.

4.5. Automatic data refresh.

In order to aid following of TJ-II operation, data can be automatically refreshed after a new plasma discharge occurs. The application receives notification of events that have occurred in the TJ-II local environment, e. g. the start or the end of a new plasma discharge sequence, or the availability of experimental data in the database for a given (in course) discharge. Hence, the user can program the automatic refreshing of time traces. In this way, when a message informing of the availability of data in the TJ-II database is received, the application automatically calls for the newly acquired data corresponding to the traces that the user has selected for automatic refresh mode. Data refresh for a given shot number can also be done manually on user demand. The user can also establish an offset for automatic refresh so that when a new signal becomes available the data that are actually called do not correspond to the last discharge, rather to the last one minus the user-defined refresh offset. This feature helps when comparing signals from consecutive discharges.

The same data refresh feature is available for all the plug-ins that could be loaded to access other data sources through the JasviMainApp interface methods (see the Multi-site access section).

4.6. Data export

The visualization application also allows exporting signal data to ASCII files that can then be read from other applications so that it constitutes a poor man TJ-II data access tool for other software packages. It exports both the header data, with information about acquisition conditions of the signal (number of bits in the ADC, number of samples, sampling period, …) followed by the time-value pairs corresponding to the signal samples.

5. Summary

New software has been developed for visual data analysis of TJ-II experimental data. This software is integrated into the TJ-II remote participation system and provides all the data visualization functionalities previously available in the local environment, as well as several new functionalities.

The visualization software is adapted to a remote participation environment and allows visualizing data from different fusion devices simultaneously by means of the integration of different data access plug-ins while maintaining the same application. TJ-II data are transferred using XML/MIME messages over the standard HTTP protocol. The use of a compressed format for the experimental data transfer allows optimizing the data transfer and reduces bandwidth necessities. In addition, the visualization application allows the use of a disk cache that reduces the access time for recently accessed data and also decreases the data server load.

The use of Java as the programming language ensures multiplatform compatibility of the software and reduced the development time. A lot of previously developed classes are available for integration with web technologies and some of which were used.

MOM software proved useful to provide the on-line synchronization of the visualization application with the local TJ-II environment and, in particular, to implement the automatic data refreshing.

Visualization software provides a very flexible and user-friendly GUI. A user can customize data plots in order to produce high quality plots. The feature of saving plot settings greatly aids a user in defining his visualization complex framework. The ability of

saving these data in a centralized server is very useful for a remote participation environment where the user is able to recover preferred settings from anywhere.

The routine developed for XY plotting of time traces has proven very efficient thus allowing very fast plots.

The PAPI security framework provides the security characteristics that are needed in a remote participation software. JNLP allows us to reduce at minimum the work related to software deployment and version control.

6. Acknowledgements

The authors thank Dr. K. J. McCarthy for his help in revising the manuscript.

7. References

[1] C. Alejaldre, J. Alonso, L. Almoguera, et al. *First plasmas in the TJ-II flexible Heliac***.** Plasma Phys. Controlled Fusion 41, 1 (1999) A539

[2] J. Vega, C. Crémy, E. Sánchez, A. Portas. *The TJ-II data acquisition system: an overview*. **Fus. Eng. and Design**, 43 (1999) 309.

[3] C. Crémy, J. Vega, E. Sánchez, C. M. Dulya and A. Portas. *Multi-processor Architecture to Handle TJ-II VXI-based Digitization Channels*. **Rev. Sci. Instr.** 70:1 (1999) 513.

[4] C. M. Dulya, C. Crémy, A. Portas, E. Sánchez, J. Vega. *Applying Object Oriented Concepts to Online Data Acquisition*. **Rev. Sci. Instr.** 70:1 (1999) 517

[5] J. Vega, C. Crémy, E. Sánchez, A. Portas, J. A. Fábregas y R. Herrera. *Data management in the TJ-II multilayer database*. Fusion Engineering and Design, 48 (2000) 69.

[6] J. Vega, E. Sánchez, C. Crémy, A. Portas, C. M. Dulya, J. Nilsson. *TJ-II data retrieving by means of a client/server model*. **Rev. Sci. Instr.** 70:1 (1999) 498.

[7] E. Sánchez, A. B. Portas, J. Vega, J. M. Agudo, K. J. McCarthy, M. Ruiz, E. Barrera, S. López. *Autonomous acquisition systems for TJ-II: controlling instrumentation with a 4th Generation Language.* **Fus. Eng. & Design** 71 (1-4) 2004, 123-127

[8] J. Vega, E. Sánchez, A. López, A. Portas, M. Ochando, A. Mollinedo, A. Sánchez, M. Ruiz, S. López, E. Barrera. *Design of the TJ-II remote participation system.* **Rev. Sci. Instr.** 74 (3) 2003, 1773-1777.

[9] J. Vega, E. Sánchez, A. Portas, A. Pereira, M. Ruiz, E. Barrera, S. López, D. Machón. *Overview of the TJ-II remote participation system.* Fusion Engineering and Design 2006 (in press)

[10] E. Sánchez, J. Vega, C. Crémy and A. Portas. *Accessing TJ-II data with remote procedure call.* **Rev. Sci. Instr.**, 72(1) (2001) 525-529

[11] J. Vega, C. Crémy, E. Sánchez, A. Portas, S. Dormido. *Encoding technique for a high data compaction in data bases of fusion devices*. **Rev. Sci. Instr.** 67:12 (1996) 4154.

[12] E. Sánchez, A. Portas, J. Vega, A. Pereira. *Applying a Message Oriented Middleware architecture to the TJ-II remote participation system.* Fusion Engineering and Design 2006 (in press).

[13] http://openjms.sourceforge.net.

[14] *JAVA Message Service*. Richard Monson-Haefel & David A. Chappell. O' Reilly. 2001

[15] M. Marinilli. *Java Deployment with JNLP and Web Start*. Sams Publishing (2001).

[16] R. Castro and Diego López. RedIris Bulletin, no. 60 (2002) (http://www.rediris.es/app/papi/doc/TERENA-2001/)

1.2.