THE TECHNICAL SUPERVISION INTERFACE: A JAVA BASED SYNOPTIC VIEW ENVIRONMENT

P. Sollander, J. Courthial, U. Epting, R. Martini, P. Ninin, C. Pesard, CERN, Geneva, Switzerland

Abstract

The development of high-level synoptic views for supervision of the technical infrastructure at CERN is becoming increasingly important. Synoptic views are traditionally used by the control room as a means of supervising remote equipment but are also becoming of increasing interest to the equipment owners themselves as the computing power and network connectivity available in offices and homes permit the use of these views. The use of common synoptic views facilitates communication between control room operators and equipment specialists and helps to limit the amount of development required. In addition, the development of synoptic views is better done by the control room operators and equipment specialist than by a computer scientist. The Technical Supervision Interface (TSI) is built to meet the new requirements of users. It will provide a Java applet viewer that can download synoptic view files from a web server and connect to any data source through a standardized protocol. The client-server communication will be done via a standardized event driven data acquisition protocol implemented on each data source type. This document describes the requirements, the design and the implementation of the Technical Supervision Interface.

1 INTRODUCTION

CERN's Technical Control Room (TCR) is today using some 700 mimic diagrams developed with a tool known as the *Uniform Man Machine Interface* (UMMI).

The UMMI tool was created in 1990 for use on control room workstations.

Over time, change requests have been made and new users like equipment groups want to use the tool. More recently, the TCR has introduced the Technical Data Server (TDS) [2] as a common data source for the different types of data clients: alarm screens, data logging and mimic diagrams. The TDS may be seen as a real-time database containing the equipment states to which the client may subscribe and receive changes as they appear.

In order to implement new user requirements, interface to the TDS properly and to prepare for the LHC, we decided start a project to replace the UMMI. A new tool should take advantage of recent visualisation technology, implement an appropriate client to the publish-subscribe mechanism of the TDS and take into account feedback on the previous tool.

2 SPECIFICATIONS FOR A NEW TOOL

User interviews were conducted with most of the current users of the UMMI and with potential users of a new system. The resulting user requirements document (URD) [3] was analysed and the following concepts were seen as important features for a new system.

- *Object-oriented graphics and symbols.* -- Object created must be completely independent and should contain everything they need to be run. Data should be device oriented rather than tag oriented.
- *Standard Symbols.* -- Many symbols; pumps, valves, etc. are the same in many views. There is a real interest in standardising these symbols largely in terms of colour use and symbol shapes. Standard symbols ease development and provide a homogeneous operational environment for the operator.
- *Flexible navigation*. -- The tool should permit totally flexible navigation in the hypertext fashion. We can forget the notion of application if we can make hyperlinks between views. Each user may set up his own "application" by organising a list of "bookmarks" to his views.
- *Event driven data acquisition.* -- This is an inherent feature of the TDS that the TSI must make use of.
- *Open architecture*. -- Not all data comes from the TDS. The TSI will define an open communications protocol that can be used by future data providers.
- *Platform independent user interface.* -- Users want to access views from control rooms, offices, their homes and firemen want access from the road. However, performance must be optimised for control room users.
- *Simplified view design.* -- The creation of a view should not need any programming. Any user should be able to design his views simply by selecting the equipment he wants to supervise from a list and place them on a screen in the way of a graphics editor with predefined symbols.

3 A NEW SUPERVISION ARCHITECTURE

The analysis of the requirements in the SRD [4] indicated the type of solution we should look into:

- Multi-platform requirements imply ***platform neutral views*** in a centralised view repository.
- The view editor needs an ***interface to a configuration database***: to have direct access to the available equipment descriptions.
- Event driven data acquisition is clearly an objective of the system. ***De-coupling the data acquisition and the data display*** will open the system to any data source, while also keeping the event driven concept for optimal performance of the user interface. The TSI must provide the necessary self-tests to ensure that the communication channels are available and notify the operator should they become unavailable.

In the Software Requirements phase of the project, we built a logical model [**Figure 1**] with the following entities:

- **View editor**: The graphics editor used to create the mimic diagrams and their dynamics.
- **View repository**: The common repository for all views.
- **Configuration database**: An off-line database holding descriptions of the equipment parameters (tags) that may be supervised by a TSI view.
- **Viewer**: The software running on the client side. It is capable of loading a view from the view repository and connecting to the necessary data servers in order to subscribe to data. It may be implemented in a web browser.
- **Data Source Adapter (DSA)**: Entity that gives access to a specific data source through a predefined standardised protocol.
- **Data Server (DS)**: Entity that dispatches messages from Data Source Adapters to the viewers and vice versa. It manages one or more DSAs and works like a data switch or a data multiplex. Its role is to permit simultaneous access to different data sources and to de-couple data source specific information (tag names, addresses) from the views.

4 THE DESIGN OF A PHYSICAL MODEL

A market survey was performed in Autumn 1998 to find a suitable commercial product for the TSI project. Six different products that fit the requirements were identified. They were products of very different size and scope ranging from complete SCADA systems including communication drivers to simple drawing editors with associated run-time software.

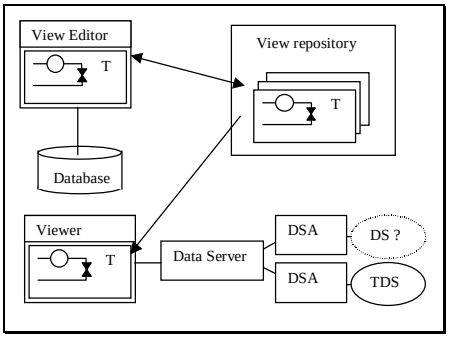

Figure 1: The logical model

The product chosen for the first TSI implementation is the product SL-GMS from the company Sherrill-Lubinski. It was chosen considering the following arguments: it has an appropriate scope (only drawing editor and run-time engine), it is a well-established product that has existed for several years and the HP-UX run-time engine is being complemented with a Java run-time engine that will be used for view access from offices and homes.

The in-house development needed for a TSI implementation using SL-GMS comprises a tag selection tool for the graphics editor, the data server (DS) and one data source adapter (DSA) for the technical data server.

There are several possible ways to distribute the above modules (Viewer, DS, and DSA) on the physical machine in the control environment. After studying the two main architectures; "Thick client", where the client application contains all modules, and the "Centralized server" where the viewer is located on the client machine and the data server on a centralized machine somewhere in the controls environment, we decided to implement the latter. It is a solution that has the following advantages (+) and disadvantages (-):

+ It makes the client smaller and lighter than the previous architecture.
+ It allows for centralized access control to data sources.
+ It permits the development of different modules in different languages (as long as they implement CORBA). This can be especially important for future implementations of DSAs.
+ It permits data filtering at the DS/DSA level to reduce the data flow to the viewer.
+ It makes it possible to have a flexible creation schema for data server and data source adapters. It may be advantageous to give priorities to different user groups or to supervise and balance the load on servers for optimal performance.
– However, it implies a centralized CORBA service that can create data servers on demand. This service must be available at all times.

For the TSI pilot implementation, we selected the "Centralized server" architecture. Below is the main class diagram for the TSI [**Figure 2**]

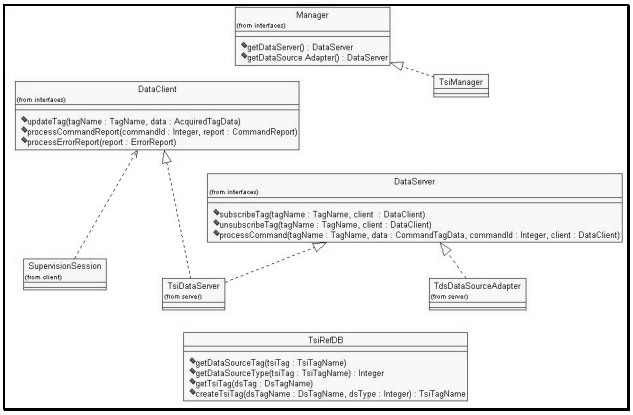

Figure 2: Main class diagram for TSI

The following entities constitute the TSI class model:

- **Manager**: This is an interface implemented by classes that create data servers and data source adapters.
- **TsiManager**: A class implementing the *Manager* interface for the TSI.
- **DataClient**: This is an interface to implement for classes requiring data.
- **DataServer**: This is an interface to implement for classes providing data.
- **SupervisionSession**: The class associated with the view. It implements the *DataClient* interface.
- **TsiDataServer**: The TSI data server class. It implements the *DataServer* interface for communication with the *SupervisionSession* and it implements the *DataClient* interface for communication with the *DataSourceAdapter* for data acquisition.
- **TdsDataSourceAdapter**: The specific data source adapter for the TDS. It implements the *DataServer* interface.
- **TsiRefDB**: The tag database for the TSI. This database stores a mapping between a data source tag name (the way an equipment parameter is defined in a data source specific context) and a TSI tag name (the way an equipment parameter is defined in the TSI context). This solution was chosen because it has the same naming conventions in all views, and also allows for easy maintenance of views.

A first prototype system has been implemented using the Orbacus CORBA implementation of Object Oriented Concepts, Inc. The *Manager*, *DataServer* and *TdsDataSourceAdapter* as well as a Java and a C++ prototype client exist at present. An interface between the SL-GMS run-time engine and the TSI *SupervisionSession* is under development and will be used in the TSI pilot application.

We have also developed a first tag selection tool which connects to the TDS reference database. This tool will be integrated with the SL-GMS editor *GMSDraw*.

5 OUTLOOK

The first TSI application will be the user interface for the CERN Safety Monitoring prototype system. It is planned to be ready by the end of 1999 and will include five views representing fire detection installations. It will use one data source adapter for the TDS.

The development of the TSI software will be iterative. After the first iteration and implementation of the basic subscription and command sending functionality, we will implement the access control and the user management features of the *Manager*.

Other DSAs will be implemented for the TSI. First in line is a DSA allowing access to archived data and functionality within the viewer to play back that data.

The *TdsDataSourceAdapter* will also be used by the *Event Logging System* [6] that will use it for data acquisition of TDS data.

It is thought that the *TdsDataSourceAdapter* could eventually be the standard CORBA interface to TDS data for any client program.

6 CONCLUSION

Finding a supervision tool for the next decade today is a difficult task. From what we have seen it looks as if WindowsNT and Java tools will dominate the market and for our use, we have opted for Java. One challenge with using Java is to provide the same performance that operators have with platform specific applications today.

We have designed the modules in the TSI to be simple, flexible and reusable and we have defined a standard interface for new data sources. It will handle other LHC data sources and could perhaps be the standard slow control data acquisition interface.

The choice of using CORBA seems the best alternative for distributed inter-object communication today. We will evaluate the performance and maintainability of it with the TSI pilot application, but we are quite confident that we have a system architecture that will last for the required period.

REFERENCES

[1] U. Epting & al "CERN LHC Technical Infrastructure Monitoring ", ICALEPCS'99, Trieste, October 1999.

[2] P. Ninin & al. "Technical Data Server: A Large Scale Supervision System", ICALEPCS'97, Beijing

[3] M-J Padilla, P. Sollander, "TSI, User Requirements Document", - CERN ST/MC/97-01

[4] P. Sollander & al, "TSI, Software Requirements Document", - CERN ST/MC/98-17

[5] P. Sollander & al, "TSI, Market Survey Report and Implementation Proposal", - CERN ST/MC/98-53

[6] M. Zurek & al, "Event Logging System, Architectural Design Document", ST/MO/99-08