AUTOMATIC CONSOLE SCREEN MANAGEMENT FOR ACCELERATOR CONTROL ROOM APPLICATIONS

P.K. Bartkiewicz, P. Duval, S.W. Herb, DESY, Hamburg, Germany.

Abstract

The number of control and diagnostic applications in accelerator control rooms has become very large. Some of the applications are running permanently, the use of others frequently depends on the accelerator state. The shift crew of the control room in DESY, Hamburg expressed a strong interest in having a console management system, driven by accelerator-related events (such as magnet cycling, injection, run, etc.), which launches appropriate applications and arranges windows on the screens. As a part of the system a remote configuration tool should be provided which offers remote access to each console computer, and permits definition of the set of applications for a given accelerator state, as well as remote start, stop and monitoring of the applications, access to local log files etc.

We describe the implementation of such a system for Microsoft Windows XP based consoles.

INTRODUCTION

The HERA lepton-proton collider requires approximately 20 operational steps, grouped into 5 procedures, to reach luminosity operation. The switching between steps, and most of operations during each of steps are largely automated and driven by a sequencer program. However, there are still approximately 150 diagnostic and control applications, which the shift crew starts on 25 consoles (PC workstations, running Microsoft Windows XP). Some of these applications need to be active for all of machine states, and some are state-specific and should be active only for a particular machine state. The complexity of management for HERA console applications in the control room is also related to the fact that physicists responsible for machine operation frequently require special programs which are written for various accelerator subsystems by different groups of engineers. There are no firm standards for application window size, and all applications can be started from any workstation, so it is important for efficiency that operator working habits and 'best practices' can be codified and reproduced so that the well known applications appear at the same console with the same windows geometry. The configurations must be easily modified and permit operator intervention when necessary. To make the preparation time for each state transition shorter the idea of a 'Console Daemon', an automatic console management tool was introduced.

REQUIREMENTS FOR THE AUTOMATIC CONSOLE APPLICATIONS MANAGEMENT

There are several requirements for such a system:

- Each console should be equipped with a server program, called here 'Console Daemon', which runs invisibly for the operator programs, but turns the console into a member of a centrally managed system.
- For the ease of maintenance and further development an existing network communication mechanism should be used; this also permits easy porting to other Microsoft Windows based consoles, for example for the HERA pre-accelerators.
- The functionality of the Console Daemon must offer the possibility of defining groups of applications which can be simultaneously started and stopped, and the positions and dimensions of windows, once established, should be preserved for the next applications starts.
- It is expected, that group definitions, applications, and windows details would be stored in local, human-readable files, but the files should be created and updated automatically by storing a 'snapshot' from the current setting of workstation, running applications, windows geometry, environmental variables etc.
- In addition, since many of control or diagnostic applications are written by non-professional programmers (mostly physicists and hardware engineers, who often write programs for handling only one specific measurement), there is a need to make a periodical 'cleanup' of workstations and to kill half-dead applications which were not properly terminated (for example applications, which have no open window, and are therefore invisible, but still have open communication links, active timers, etc., and consume console resources).
- The system should be driven by the existing HERA sequencer program, but a remote configuration and management program is also required, as well as simple command line programs, which might be used in scripts and batch files.

IMPLEMENTATION OF THE AUTOMATIC CONSOLE APPLICATIONS MANAGEMENT

The system consists of three programs: the *Console Daemon*, which is installed on each console workstation, the *Remote Console Screen Manager*, an application providing a remote access to all Console Daemons, and supporting the same functionality as the local Console Daemon popup menu, and the *Command Line Interface*, a simple program '*setconsmode*', which is intended to be used in scripts or batch files.

Implementation of the Console Daemon

The Console Daemon was written in Visual Basic 6 with very strong use of the Win32 API (mostly related to functions from kernel32.dll, user32.dll, advapi32.dll, shell32.dll, ps.dll). [4,5,6] The server functionality was implemented by embedding an ActiveX server for TINE (the RPC-based communications protocol used for the HERA control system). [1,2] In order not to occupy space on the Windows task bar or the desktop, the daemon icon appears only in the system tray (fig.1).

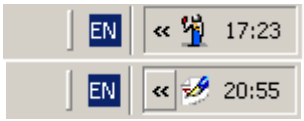

Figure 1: Icons of Console Daemon, installed in the system tray: operational mode and editing mode.

Immediately after installation, the Consol Daemon is recognized by the control network as a TINE server and can handle remote requests. At the same time the Console Daemon can be operated locally; the user can request a pop-up menu (fig.2) by clicking on the program icon. All of the functionality available as menu selections is also accessible through the server interface.

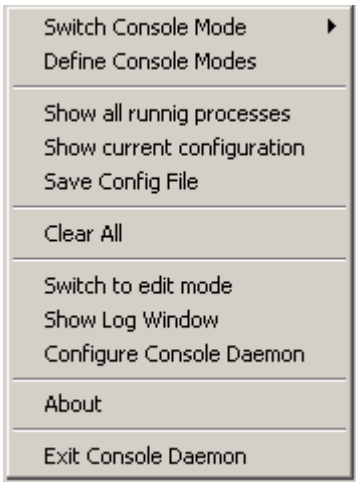

Figure 2: The Pop-up menu, available after clicking on Console Daemon's icon.

The Console Daemon has two working modes: operational mode and editing mode.

The Operational Mode

The operational mode is the default mode, which is activated after the start of the Console Daemon. Just after entering operational mode Console Daemon reads its locally stored XML configuration file. This file contains definitions of application groups, which are identified as working modes of the console. For each console mode groups of applications are listed, and for each application the name, path, command line arguments, window captions, and geometry are stored. The first console mode listed in the configuration file is chosen and become 'active'.

On selecting a console mode, all applications previously started which do not belong to that mode are stopped, and all other applications belonging to the mode are started. After a given period of time, windows are placed in desired locations and resized to specified dimensions. The geometry of windows can not be set immediately after the applications start, since many applications execute a relative long self initialization stage, so that their windows are popped up some seconds later. Optionally, for very important applications, an automatic restart is foreseen, and as another option, windows geometry and locations can be kept during the lifetime of the application. Switching to another defined console mode is possible at any time by sending an RPC request from the network client application or by the operator selecting the appropriate item from the pop-up menu. The switch to the editing mode can be done locally, by a selection on the pop-up menu, or remotely, using a dedicated remote configuration tool.

In addition it is periodically checked whether all applications have visible windows, to catch improperly terminated applications which have lost their windows, but still consume system resources. For that purpose the list of all available control applications is read, and any application listed there which is found to be running without a window, is, after repeating the check several times, killed (the application might have just been started and not had enough time to pop up the window). This mechanism cleans the system and guarantees that all system resources are freed.

The Editing Mode

The editing mode is used to define console modes and to build the group of applications for each console mode, specifying arguments for applications and defining windows geometry. Although the configuration is stored in the XML file, the entire configuration process can be done without the need to edit any files. The typical editing session proceeds as follows: after switching the Console Daemon to editing mode the user may create a new mode by making a selection on pop-up menu and typing a name of the mode. The newly created mode is added to the list of existing modes and becomes available on the pop-up menu. The user chooses the mode to be edited, stops

applications which will not belong to that mode, starts the applications desired for the mode, and locates and resizes their windows. When the setup of applications is complete, the user requests a save to the configuration file. The Console Daemon then builds a list of running applications, and queries the system for the information about command line arguments which were supplied to the running applications, starting paths, parent processes, and geometry and caption of each of opened window. These data are stored in locally in the XML configuration file.

Configuring the Console Daemon and some tools functions

There are some parameters which can be set for the Console Daemon (fig. 3): how frequently the process list should be refreshed, the delay between application start and window positioning, and whether applications should be started directly or via a launcher script. Additionally for some crucial for accelerator operations workstations, auto-restart of applications can be selected, as well as keeping windows in specified positions. If the options should be made persistent, the user requests a save to the XML configuration file.

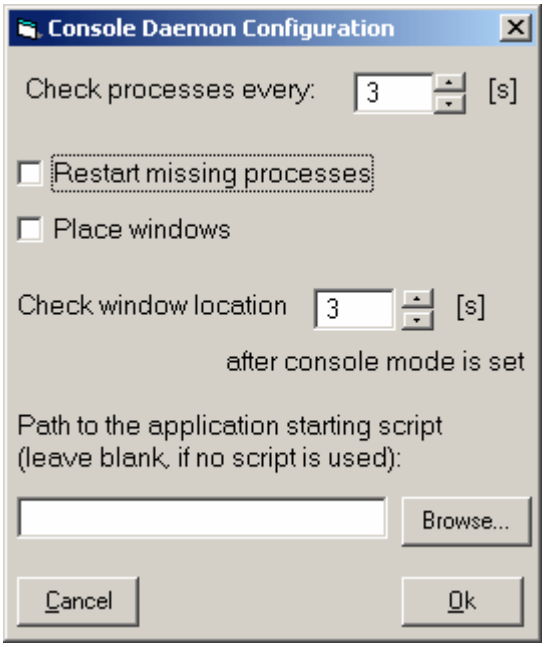

Figure 3: The Console Daemon configuration window.

The Console Daemon creates a locally stored, one day log file; the most recent messages can be viewed by selecting the appropriate menu item. Some of these messages can help authors of applications to learn whether their applications crashed. For such debugging purposes Console Daemon offers a window with useful process information, much of which is not available from the standard Microsoft Windows Task Manager..

Implementation of the of the Remote Console Screen Manager

The Remote Console Screen Manager (RCSM), also written in Visual Basic 6, is a network client with which users can connect to any of Console Daemons in order to get the same functionality as that offered by the Console Daemon local menu. The TINE protocol is implemented with the ACOP ActiveX control supplied by the TINE control system. After the RCSM starts, it gets from the TINE name server a list of the Console Daemons available in the system. It then queries each Console Daemon for the information about its configuration and defined console modes and the corresponding applications, and builds the tree-list, shown in the left part of the user interface (fig.4). When a user selects the console by clicking a tree-list item, the RCSM sends the request to the appropriate Console Daemon and obtains the system specific information (such as number of screens and resolution of each screen), a screen shot, and information for all running processes. The screenshot might be presented in 1:1 scale, or fitted to the box. One and multi- (currently up to 4) screen consoles are supported. The screen shot can be refreshed at any time by user request, as can information about the processes. Information about the selected process and its windows is presented in the upper part of the application.

Besides the monitoring functionality, the RCSM provides users with the possibility of starting or stopping any application available on the network file server. It can be used to define and select the console mode and, using movable rectangle shown on the screen shot, to change the geometry and position of each window. After the changes are done, user can remotely request the save of the XML configuration file.

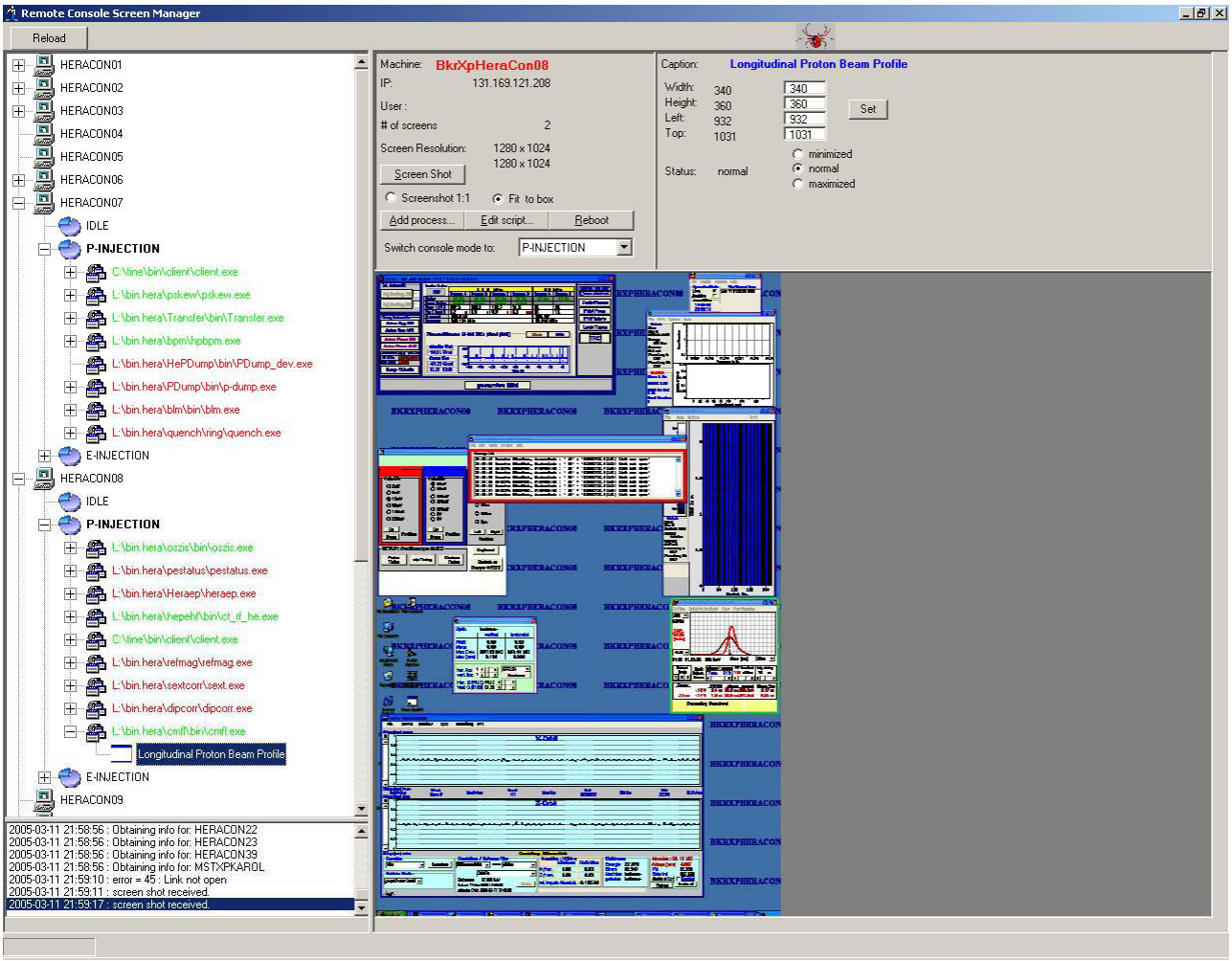

Figure 4: The Remote Console Screen Manager, connected to the dual screen console.

Implementation of the Command Line Interface

In order to make possible control of Console Daemons by scripts and desktop shortcuts, a simple command line application was written in the 'C' language. The application accepts two parameters: the name of the Console Daemon server (which must be a name recognized by the TINE name server), and a string identifying a requested console mode. A sample script switching the modes for two consoles might contain the following lines:

```
setconsmode  CD_HERACON08 e-injection
setconsmode  CD_HERACON09 e-injection
```

STATUS AND FUTHER PLANS

The Console Daemons have been installed on all HERA console workstations. For three of the consoles sets of scripts and associated desktop shortcuts for switching console modes were created; these are used by the shift crew to set up consoles for a particular machine operation with a (double) mouse click. A successful test of controlling consoles from the HERA sequencer has also been made. The operational use of sequencer to drive Console Daemons is being considered for the run period after the next longer maintenance shutdown of HERA, in order not to disturb current luminosity operations.

The Remote Console Screen Manager is still in the debugging phase.

REFERENCES

[1] http://desyntwww.desy.de/tine/
[2] Philip Duval, "The TINE Control System Protocol: Status Report" Proceedings PCaPAC 2000, 2000.
[3] J. Maass, "Sequencing and Ramping in Hera" PCaPAC'99, KEK, Tsukuba, Japan, January 1999
[4] Daniel Appleman, "Visual Basic Programmer's Guide to the Win32 API", Ziff-Davis Press
[5] http://msdn.microsoft.com/library/
[6] http://www.activevb.de