PROTON THERAPY TREATMENT ROOM CONTROLS USING A LINUX CONTROL SYSTEM

J.Katuin
Indiana University Cyclotron Facility, Bloomington, IN 47408, USA

Abstract

The Indiana University Cyclotron Facility (IUCF) is in the process of completing the building of the Midwest Proton Radiotherapy Institute (MPRI). The design of MRPI's proton therapy system required the development of several control systems responsible for delivery of proton beam to a patient and patient positioning. One such control system is the Treatment Room Controller. This system allows for management of the other control systems, and is the primary user interface to the proton therapy system. This control system was developed with a Linux operating system, the KDE/QT widget sets for the user interface, and the KDevelop IDE. The control software uses the unixODBC API to provide an interface to a MySQL database for record and verify functions and for history functions. The control system also uses the Comedi driver library so that a National Instruments PCI DAQ card can be used to interface to various treatment room devices.

MPRI PROTON THERAPY SYSTEM CONTROLS DESCRIPTION

The MPRI Proton Therapy System (PTS), as shown in Fig. 1, is designed to provide proton therapy using two primary control lines, the Beam Delivery Line and the Patient Handling line. Both lines contain sub-systems with their respective control systems that together provide proton therapy to a patient. Such systems are grouped into two categories, Beam Delivery, and Patient Handling [11].

The Beam Delivery Group is responsible for proton beam delivery, dose monitoring, and radiation safety. This group is composed of the Beam Delivery System (BDS), the Dose Deliver System (DDS), the Kicker Enable System (KES), and the MPRI Interlock and Radiation System (MIRS).

The Patient Handling Group is responsible for patient positioning and position verification of a target. This

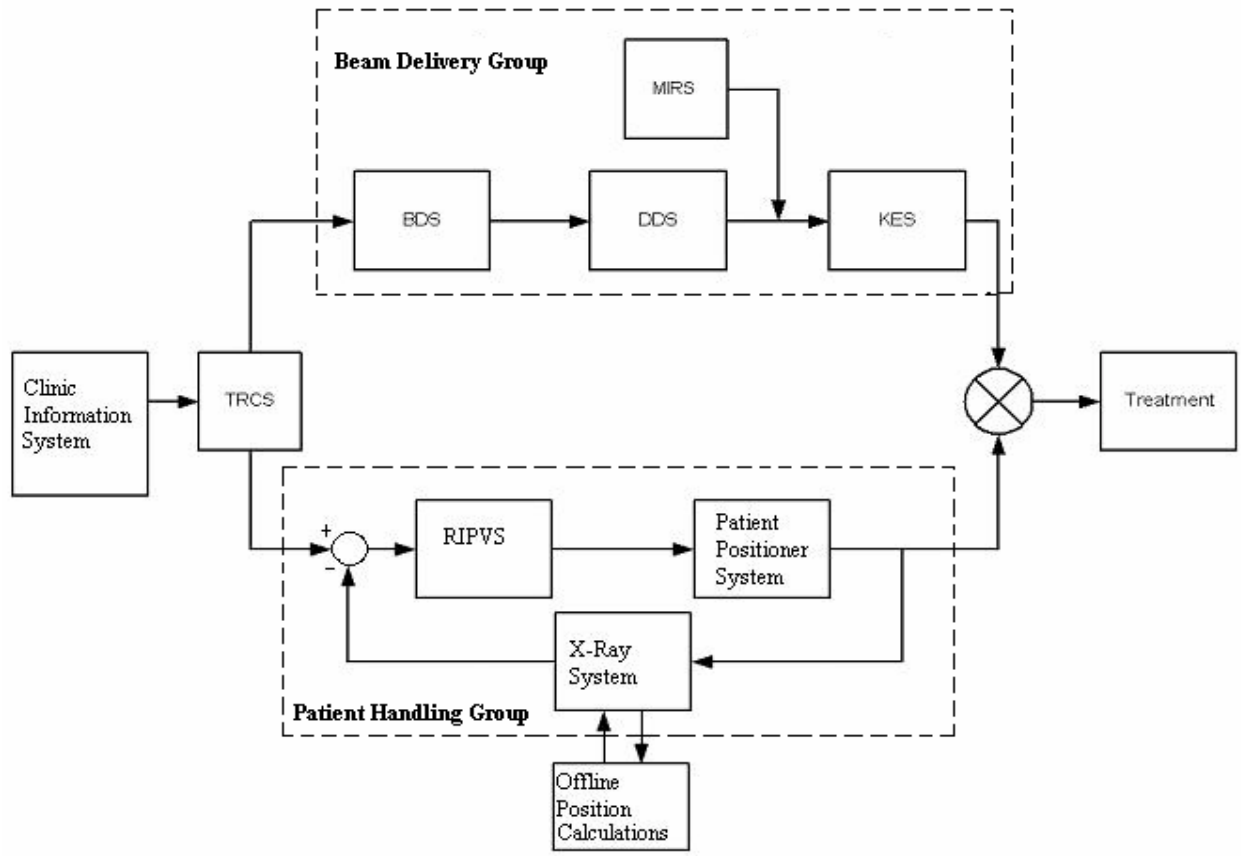

Figure 1: The Proton Therapy System.

group is composed of the Patient Positioning System (PPS), the Robot Interface and Position Verification System (RIPVS), and the X-Ray System.

The Treatment Room Control System (TRCS), an Intel Pentium based computer system, provides clinic personnel with the primary user interface for the PTS system for implementing and managing the treatment process. The TRCS accomplishes this by providing the interface with the Clinic Information System for treatment requirements and results.

When a treatment process is initiated, the TRCS retrieves the Treatment Planning Package from the Clinic Information System. This package contains treatment specific parameters, instructions, and files used for treatment by the various systems in the PTS. The TRCS downloads the elements of the package to the respective systems, and then verifies the information for correctness. Upon completion of a fraction the treatment events are sent back to the Clinic Information System in the form of a History Package, including x-ray images, log files, and corrections.

The TRCS also inspects the installation of treatment specific devices such as apertures, boluses, and ridge filters that are associated with a patient using bar code identification and limit switches. The TRCS will also display nozzle position by providing a signal from a potentiometer mounted to the nozzle assembly. The TRCS will monitor energy setup by analyzing hall probe data from a switching magnet located in the Beam Delivery System. This monitoring will allow for an independent check on the energy setup in the BDS.

OPERATING SYSTEM AND SOFTWARE DESIGN

Operating System

A requirement of the control system design was to use a Linux operating system so that during the development phase of the project there could be an X- windows mechanism for monitoring other PTS control systems that had X-server capabilities. Also, a Linux operating system provides flexibility during the development of the TRCS by allowing remote logins, free development tools, and a relatively stable multitasking environment for applications. The Linux operating system needed to conform to an accepted standard. The intent was to ensure that the kernel release would allow for simple implementation of open source components such as the unixODBC and Comedi drivers. Consequently, SuSe 8.1 [1] was chosen for the operating system since it follows the United Linux Standard [2].

Software Design

The software requirement for the TRCS was to provide a GUI application called the "Treatment Room Manager" (TRM). The architecture of this software, as shown in Fig. 2, was to be divided into 5 groups, Patient Data Group, which handled patient/treatment information in the form of Treatment Package and History Package, Treatment Management, which handles direct therapy treatment commands from the Radiotherapy Technologist (RTT), the Maintenance Group, which allows for testing and configuration of the system, the Communications Group, which manages the communications and treatment requests to and from the other PTS systems using TCP/IP sockets, and the DAQ Group, which handles the analog and digital input/output signals, as well as control logic.

The application GUI uses the KDE and QT widget sets [3][4]. Consequently, the user interface was designed with several user interface screens. TRM uses an Explorer Layout, as shown in Fig.3. Access to such screens is controlled through user specific privileges.

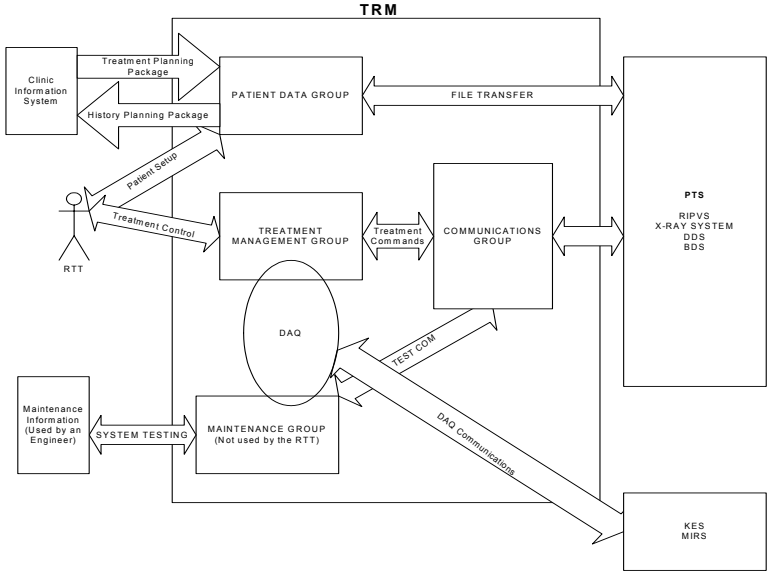

Figure 2: Software Architecture.

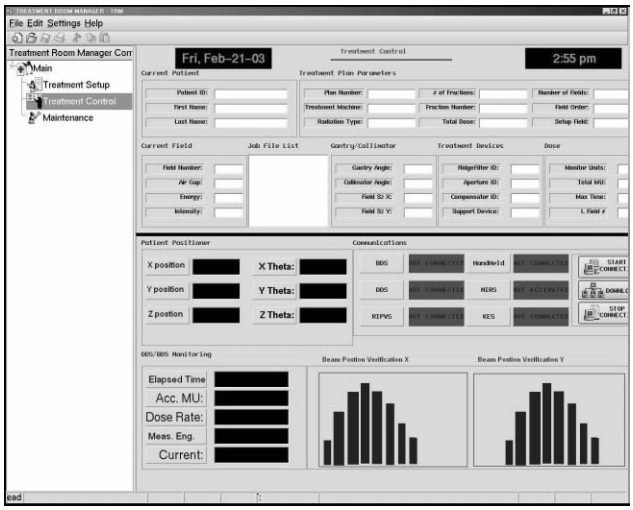

Figure 3: TRM GUI screen for the Treatment Management.

Since the application is written in C++, the QT and KDE widget were easily implemented. Also, the QT and KDE widget sets provide a signal/slot mechanism, which allows for simple inter-process and event driven communications [5].

KDevelop 2.1[6] was chosen as the development environment because it provides open source application-building tools that allow for rapid development of X-windows programs. Since the TRM application uses QT widgets for each screen, basic screen layout was accomplished using QT Designer [7], and then the widgets were added to the TRM project for compilation.

Database Interface

Early in the development of the TRM the actual database server was undefined, so a more general solution to the database interface was sought. Consequently, unixODBC [9] was chosen so that the code would require little to no changes regardless of the database server platform. Currently, the TRM interfaces to a MySQL Server in the MPRI Clinic Information System. Although the unixODBC web site provides some basic code examples, it lacks information on all of the API calls available to the unixODBC driver. However, this driver is compatible to Microsoft's ODBC references found at Microsoft's MSDN web site under Data Access[10]. Consequently most ODBC API calls described by Microsoft are applicable to the unixODBC.

Data Acquisition and Control

Data Acquisition and control for reading the magnet hall probe, nozzle potentiometer, and various limit switches was accomplished using a National Instruments PCI-6025E card. This card was chosen for the number of digital I/O channels, and analog channels. National Instruments recommends the use of the open source Comedi driver [8]. This driver package provides libraries for both real-time and non real-time Linux releases and can be used with a wide range of DAQ card manufactures.

The TRCS required both sets of libraries for the control application. The non real-time library (Comedilib [8]) was used for TRM because the DAQ and control requirements are non-critical, and the TRM is a user space application. A time critical feature was specified as a health signal to the KES, which oscillated at a fixed frequency of 10 hertz. If KES does not receive the signal then beam delivery into the treatment room will stop.

A kernel module was written using the real time library (Kcomedilib [8]). This module received a software health signal from TRM through a Char driver file [12], which allows user space/kernel space application communication. The TRM processes that need verification of operation were summed in intervals that were acceptable to the kernel module, which in turn allowed the kernel module to continue outputting a signal to the KES. The significance of this arrangement was that both drivers could communicate with a single NI card, and that the timing output of the kernel module was a consistent 10 hertz. If the TRM is disabled for any reason then the health signal will stop.

CONCLUSIONS

The TRCS has been designed using Linux and open source resources and standard hardware to meet the clinic requirements of MPRI's PTS. The tools are available at a minimal to no cost to developers of such a system. The development environment available with KDevelop and the KDE and QT widget sets allow for relatively fast application development. The unixODBC driver provides a means by which a programmer can create database applications that are server platform independent. The Comedi driver package provides a means of programming DAQ cards in a Linux environment, including kernel level applications. Support for this work is provided by the State of Indiana, Indiana University, the DOE (Grant No. DE-FG-02000ER62966) and the NIH (Grant No. CO6 RR17407).

REFERENCES

[1] http://www.suse.com
[2] http://www.unitedlinux.com
[3] http://www.kde.org
[4] http://www.trolltech.com
[5] http://www.trolltech.com/products/qt/whitepaper
[6] http://www.kdevelop.org
[7] http://www.trolltech.com/products/qt/designer.html
[8] http://www.comedi.org
[9] http://www.unixODBC.org
[10] http://msdn.microsoft.com/library/
[11] http://www.ni.com/linux/daq_comedi.htm
[11] J. Katuin & A.N. Schreuder, Proc. CAARI 2002, Denton, TX (2002) to be Published.
[12] Rubini & Corbet, "Linux Device Drivers, 2nd Edition", O'Reilly, June 2001