EPICS-BASED CONTROL SYSTEM FOR A RADIATION THERAPY MACHINE

Jonathan Jacky*, University of Washington Medical Center, Seattle, WA 98195, USA

Abstract

The clinical neutron therapy system (CNTS) at the University of Washington Medical Center (UWMC) has been treating patients since 1984. A new EPICS-based therapy control program replaces a locally-developed C program used since 1999. The new program retains the original safety philosophy and delegation of functions among nonprogrammable hardware, PLCs, microcomputers with programs in ROM, and general-purpose computers running high-level language programs. The latter are used only for data-intensive, prescription-specific functions. The therapy control portion uses a single soft IOC for control and a single EDM session for the operator's console. Prescriptions are retrieved from a PostgreSQL database and loaded into the IOC by a custom client program. The system remains safe if the computers or their programs crash or stop producing results. Different programs at different stages of the computation check for invalid data and other faults. Development activities include formal specifications and automated testing which avoid, then check for, design and programming errors.

INTRODUCTION

The Clinical Neutron Therapy System (CNTS) at the University of Washington Medical Center (UWMC) has been treating patients and making isotopes since 1984 [1]. The system includes a cyclotron and a treatment room with an isocentric gantry and leaf collimator operated under computer control (far ahead of its time in 1984). The system was built and installed by a vendor, but since then has been maintained and upgraded by UWMC staff. In 1999 we replaced the therapy portion of the vendor's original control system (a PDP11 programmed in FORTRAN) with new hardware and our own software (a 68040 in a VME crate running VxWorks programmed in C) [2].

We are now replacing the therapy control system once again. The goals are to retain all the present therapy functionality (initially), to avoid obsolescence by using a modern, supportable hardware/software platform, to accommodate changes in the surrounding environment (especially hospital information systems for treatment planning and record keeping), and possibly to support new therapy techniques, for example intensity-modulated radiation therapy (IMRT).

Meanwhile, we have gained several years of experience programming and running the EPICS control system framework for non-therapy functions of the cyclotron and beamlines. We would like to use EPICS for therapy control as well, in order to take advantage of its many features and to standardize on a single collection of long-lived and widely used components and development tools.

EPICS has been used for over twenty years in many demanding applications [3, 4, 5]. Nevertheless, experienced EPICS developers have reported that they do not use EPICS to implement safety-critical controls, but only to monitor them [6, 7]. We can find no reports of EPICS used in safety-critical controls.

Our therapy control system also uses non-EPICS components for safety-critical functions nearest the hardware. However, we do use EPICS to process some of the data that is input to these functions, and to process output that is collected for record keeping. So EPICS is on some safety-critical signal paths. These processing steps are only feasible for high-level language programs running on general-purpose computers. They require both careful analysis and assurance during development and intensive monitoring during operation — whether or not they use EPICS.

EPICS is not an application, it is a framework or toolkit with many optional components that developers can use to build applications. It is not meaningful to say that EPICS itself is safe (or not). It is only possible to evaluate a particular application (including some EPICS components) that must satisfy particular requirements running in a particular environment. In this report we begin to sketch how we do this for our neutron therapy control application. In order to do this, we must first describe its function, safety requirements, and composition.

THERAPY MACHINE DESCRIPTION

The cyclotron produces a proton beam that strikes a beryllium/copper target, producing a broad neutron beam. (The control systems for the cyclotron and beamline upstream from the target also use EPICS but are not considered in this report.) The shape of the neutron beam that reaches the patient is determined by setting forty steel collimator leaves (to conform to the shape of the tumor). The dose within this aperture is modulated radially by interposing one of two flattening filters (always) and may be further modulated in one direction by interposing one of several wedge filters (optionally). The entire collimator including the leaves and filters is mounted on a gantry that rotates 360 degrees so the beam can enter the patient from above, below, or any other angle. The collimator assembly rotates around the beam central axis so the leaves can be oriented advantageously. The wedge filters rotate independently within the collimator. The patient lies on a couch with five degrees of freedom (three linear motions and two

*jon@uw.edu. Thanks to the UWMC cyclotron engineering group: Stefani Banerian, Eric Dorman, Robert Emery, Steven Steininger, and Tobin Weber

rotations) so the beam can enter the patient from almost any direction.

To treat a patient, the patient and couch are positioned and all moving components (including the filters and all forty leaves) are set to their prescribed positions, then all motions are disabled. The cyclotron RF drive is turned on, which turns on the proton beam and the neutron beam. A dosimetry system integrates the accumulating dose from the neutron beam until the prescribed dose is reached. Then the dosimetry system turns off the cyclotron RF drive, so the proton and neutron beams turn off.

A treatment beam prescription is specified by the settings of all moving components (couch, filters, leaves), and by the daily prescribed dose and the number of daily treatment sessions (which determine a total prescribed dose). The collection of beam prescriptions for each patient is stored in a prescription database. (Preparing and checking the prescriptions in this database involve a major quality assurance effort, not discussed in this report.) There is also a treatment record database that records the treatments that are actually delivered, for ongoing patient quality assurance, and recovery if a treatment ends prematurely.

Treatments are performed under the control of human operators, not the cyclotron operators or engineers, but other hospital staff: radiation therapy technologists. Technologists identify each patient and select each prescribed beam from the database. They position the patient on the couch and manually (by hand and eye) set all external motions (gantry rotation, collimator rotation, and all five table motions). They command the control system to set up the internal motions (filters and leaves) and initialize the dosimetry system with the prescribed dose. They command the beam to turn on.

SAFETY REQUIREMENTS

The primary safety requirement is that the neutron beam can only turn on or remain on when all the settings conform to their prescribed values, and the daily dose and total dose are less than prescribed. The control system enforces this requirement. For example, if a technologist attempts to turn on the beam when a motion setting is not in the prescribed position, the beam will not turn on. If a setting moves out of the prescribed position (even if the motion just becomes enabled) when the beam is on, the beam will turn off.

CONTROL SYSTEM COMPONENTS

Wherever possible, therapy controls are implemented with nonprogrammable elements (relays etc.), programmable logic controllers (PLCs), or embedded microcomputers with programs in read-only memory (ROM). Turning the beam on or off, and moving components under manual control, are implemented in this way.

However, some therapy control functions are only feasible for high-level language programs running on general-purpose computers. These functions use the (voluminous, complex) data in the prescriptions (which are different in each prescription). These functions include retrieving prescriptions from the database, loading the prescribed settings into the low-level device controllers, comparing the actual settings read back from the controllers to the prescribed settings, commanding controllers in the proper sequence, and storing treatment records in the database.

An EPICS application comprises one or more Input/Output Controllers (IOCs) and client programs (that provide user interfaces and data management), which communicate using a network protocol, Channel Access (CA). An IOC can be a dedicated embedded computer, or a "soft IOC", a program running on a general-purpose computer. The application code that runs on an IOC is called a "database", actually a data flow program defined by connecting predefined processing elements called "records".

Our therapy control program runs on a single soft IOC, the only application program on a dedicated x86 PC running Linux. Its database includes records to store all the prescribed settings and all the actual settings read back from the attached device controllers, and records to compute the readiness to turn on the beam, broken down by subsystem and setting. This database also includes records to command the controllers and sequence their activities.

During normal operation the therapy control computer uses TCP/IP over Ethernet to communicate with just four other components: the database server, the therapy technologist's console, a PLC, and a serial port server.

The database server is a Linux PC running the PostgreSQL database [8], which stores both the prescription database and the treatment record database.

The technologist's console is a Linux PC running an EPICS client program, the Extensible Display Manager (EDM) [9], attached to a large touch-screen display. EDM provides the graphical user interface (GUI) to the therapy control program, that technologists use to select prescriptions, to monitor therapy machine status (its readiness to turn the beam on, etc.), and to command the control system to automatically set up leaves, filters, and dosimeters. When a technologist selects a prescription, EDM invokes a client we wrote, a Python program that uses the `psycopg2` library [10] to retrieve the prescribed settings from the database server and then uses the `pyepics` CA library [11] to load them into the IOC.

The PLC handles most single-bit digital input/output (mostly relays) including sensing a nonprogrammable Hardware Safety Interlock System (HSIS). The PLC and HSIS include safety and consistency checking that are independent of the therapy control computer. The therapy control computer communicates with the PLC over the Ethernet using the EPICS `asyn` device support [12] with the `modbus` driver [13].

The serial port server connects via RS232 to three microcomputers programmed in ROM, which provide device control: the Treatment Motion Controller (TMC), Leaf Collimator Controller (LCC), and Dose Monitor Controller (DMC). These in turn connect to nonprogrammable hardware with built-in safety mechanisms. For example, once the prescribed dose is loaded into the dosimetry system, the dose is integrated and the beam is turned off by hardware (not the microprocessor program). The therapy control computer communicates with the these controllers through the serial port server using the EPICS `asyn` device support with the `StreamDevice` driver [14].

ASSURANCE DURING DEVELOPMENT

A safety-critical system requires both careful analysis and assurance during development and intensive monitoring during operation. In this section we discuss development.

Safety analysis

The facility has operated without mishap for almost thirty years. Although some components are new, the overall control system functionality and design including the delegation of functions among the therapy control computer, PLC, HSIS, TMC, LCC and DMC remains the same. The strongest evidence for its soundness is its successful performance history.

A safety analysis is an explicit description of potential hazards and countermeasures. The original vendor did not provide one to us. We did do some *post hoc* analysis, in effect justifying the rationale for the system as built (summarized in [2]). This should help prevent us from inadvertantly introducing new hazards. The most important findings are that the system exhibits a conservative design which eliminates some hazards and removes others from computer control. For example:

The system can always achieve a safe state without any computers by turning off the beam and disabling motions. There is no need for redundancy to ensure the computers keep running.

Guiding external motions and avoiding collisions is entirely delegated to human operators. External motions are disabled by nonprogrammable hardware except when explicitly commanded by an operator.

Only one type of radiation (neutrons) at one energy can be produced in the treatment room. The dose rate is low enough so that negligible dose is delivered after the beam is commanded to turn off.

The therapy control program has no hard real-time deadlines, those are all delegated to the HSIS, PLC, and microcomputers.

The HSIS, PLC, and each controller can maintain or achieve a safe state for the components they control, even if other computers are not running.

If a computer or controller stops or behaves erratically, a watchdog timer detects this and places the system in a safe state.

And so on. Most hazards that implicate the therapy control program are limited to loading the prescribed settings and checking that the actual settings conform. We concentrate on these in the rest of this report.

Functional specification

It is necessary to have a comprehensive description of the intended behavior to guide coding, test planning, and safety analysis. The internals are new, but the behavior of the new EPICS program is almost identical to our 1999 program [2] so some of the products of that development are still valid. These include a 240-page reference manual that now serves as our functional specification [15], and a formal specification in a mathematical notation, that describes the program as a collection of state transitions and expresses the safety invariants that must be maintained by each transition [16].

EPICS programming style

We use as few EPICS features and mechanisms as possible, in order to keep our program easy to understand and (we hope) reliable. The entire therapy control program runs on a single soft IOC, the only application program on its computer. The IOC is self-contained in the sense that it does not need to communicate with any clients to achieve or maintain a safe state. The entire IOC program is expressed by EPICS database records, `StreamDevice` protocol files, and the IOC startup command file `st.cmd`. There is no custom device support nor any other custom code, not even subroutine records. We do not use the EPICS State Notation Language or the sequencer. The IOC only uses the EPICS Channel Access (CA) network protocol to load prescribed settings when a prescription is selected, and to communicate with the EDM client. The database itself uses no CA links, only local database links. All control flow in the database is "pushed" from input to output, using `SCAN PASSIVE`, `OUT PP`, and `FLNK` fields.

We use these record types: `acalcout, ai, ao, asyn, bi, bo, calc, calcout, fanout, longin, longout, mbbo, scalcout, stringin, stringout, seq`.

Inspection and review

Two kinds assurance activities can be done during development: static analyses where the code is not executed (including reviews and inspections) and dynamic analyses (testing).

To help us review our own code, we created a tool based on the `dot` graph layout package [17] that generates annotated data-flow and control-flow diagrams from EPICS databases.

We study the available documentation for all the EPICS components we use (enumerated in the preceding sections), including discussion on the `Tech-talk` mailing list.

It would be salutary to look at the EPICS core code and device support code that we use, along the complete signal path from input through the database to output. We limited our use of EPICS components to make this more feasible.

Testing

We use the Python `unittest` module [18] to script automated tests for the therapy control program. Most test scripts use the pyepics library [11] to read and write process variables (PVs) in the IOC, so they bypass the device support but test the program logic.

Some scripts perform stress tests. For example, we tested the signal path from the prescription database to the IOC, through psycopg2 and pyepics. In about thirty minutes, we retrieved different sets of prescribed settings from from the database ten thousand times, each time comparing the resulting IOC contents against the original database contents. (A typical rate is less than ten.) There were no errors.

MONITORING DURING OPERATION

Development activities including code reviews and tests provide confidence that the therapy control program is logically correct. But perhaps it is possible for disturbances in the environment or rare transient conditions to interrupt or delay program execution in ways that might be difficult to detect (despite the watchdog timers). What if some, but not all, settings are updated when a prescription is retrieved? What if some, but not all, database calculation records stop processing when prescribed settings and actual settings do not agree?

We are considering various run-time checks for such occurences. For example, to guard against corruption of retrieved prescriptions, we could store a checksum of the prescription in the database, and compare it to the same checksum computed by the IOC from the retrieved settings.

PROJECT STATUS

At this writing, the new therapy control system is not complete and has not yet begun clinical operation, but enough is working to permit some preliminary conclusions.

CONCLUSION

We expected that the greatest risk of switching from C on VxWorks to EPICS on Linux would arise from the size and complexity of the latter [19, 20], which we feared might reduce reliability and make behavior more difficult to analyze and predict. Our experience so far indicates that in our application, EPICS is as trustworthy as our previous platform, and provides other advantages as well. Additional inspection, testing, and runtime monitoring remain to be done to confirm this impression.

REFERENCES

[1] R. Risler, S. Banerian, J.G. Douglas, R.C. Emery, I. Kalet, G.E. Laramore, and D. Reid. 25 years of continuous operation of the Seattle clinical cyclotron facility. In Youjin Yuan, Lina Sheng, and Lijun Mao, editors, *Proceedings of the Nineteenth International Conference on Cyclotrons and Their Applications*, pages 68–70, 2010.

[2] Jonathan Jacky, Ruedi Risler, David Reid, Robert Emery, Jonathan Unger, and Michael Patrick. A control system for a radiation therapy machine. Technical Report 2001-05-01, Department of Radiation Oncology, University of Washington, Box 356043, Seattle, Washington 98195-6043, USA, May 2001. http://staff.washington.edu/jon/cnts/cnts-report.html.

[3] L. R. Dalesio, M. R. Kraimer, and A. J. Kozubal. EPICS architecture. In C. O. Pak, S. Kurokawa, and T. Katoh, editors, *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, pages 278–282, 1991. ICALEPCS, KEK, Tsukuba, Japan.

[4] Matthias Clausen and Leo Dalesio. EPICS - Experimental Physics and Industrial Control System. *Beam Dynamics Newsletter*, (47):56–66, Dec 2008. http://www-bd.fnal.gov/icfabd/Newsletter47.pdf.

[5] http://www.aps.anl.gov/epics/.

[6] Simon Rees and Andrew Johnson. Re: Is there anyone using EPICS for a FDA approved device? Tech-talk mailing list. Tue, 19 Aug 2008 10:56:21 +0200, http://www.aps.anl.gov/epics/tech-talk/2008/msg00803.php.

[7] Andrew Johnson. Re: Doubt regarding standards followed by EPICS. Tech-talk mailing list. Mon, 10 Sep 2012 16:52:29 -0500, http://www.aps.anl.gov/epics/tech-talk/2012/msg01836.php.

[8] http://www.postgresql.org/.

[9] http://ics-web.sns.ornl.gov/edm/.

[10] http://initd.org/psycopg/.

[11] http://cars.uchicago.edu/software/python/pyepics3/.

[12] http://www.aps.anl.gov/epics/modules/soft/asyn/.

[13] http://cars9.uchicago.edu/software/epics/modbusDoc.html.

[14] http://epics.web.psi.ch/software/streamdevice/doc/.

[15] Jonathan Jacky and Ruedi Risler. Clinical neutron therapy system reference manual. Technical Report 99-10-01, Department of Radiation Oncology, University of Washington, Box 356043, Seattle, Washington 98195-6043, USA, October 1999. http://staff.washington.edu/jon/cnts/refman.html.

[16] Jonathan Jacky. Formal safety analysis of the control program for a radiation therapy machine. In Wolfgang Schlegel and Thomas Bortfeld, editors, *The Use of Computers in Radiation Therapy: XIIIth International Conference*, pages 68–70, Berlin, Heidelberg, New York, 2000. Springer. http://staff.washington.edu/jon/cnts/iccr.html.

[17] http://www.graphviz.org/.

[18] http://docs.python.org/2/library/unittest.html.

[19] Nick Rees. EPICS internals. Slide deck, 2010. http://controls.diamond.ac.uk/downloads/other/files/EpicsInternals.pdf.

[20] Mark Rivers, Marty Kraimer, and Eric Norum. asyn: An interface between EPICS drivers and clients. EPICS Collaboration Meeting, April 2012. https://portal.slac.stanford.edu/sites/conf_public/epics_2012_04/presentations/asynTalk.pdf.