Using Formal Models to Plan Control System Testing

Jonathan Jacky
Alexandra Barke *
Ruedi Risler

Radiation Oncology Department RC-08
University of Washington
Seattle, WA 98195

jon@radonc.washington.edu

Technical Report 91-02-02

February, 1991

Abstract

We modeled and tested an embedded controller that will be incorporated into a medical process control application. The controller (including customized software) was developed by others, and its original documentation does not include a functional specification that is sufficiently complete and accurate for our needs. We proposed a functional specification of certain important controller behaviors, based on the available documentation and behavior that we had observed. We used Petri nets for our specification notation. From the Petri net, we derived a series of tests intended to reveal whether our provisional specification accurately characterized controller behavior. The method for deriving the tests from the Petri net is described. The tests systematically traverse the Petri net and also present faults that should be handled in a reasonable way. Performing the tests revealed several errors in our initial specification. We modified our specification until it was consistent with all observed behaviors, covering a wide range of situations.

*Department of Computer Science and Engineering, FR-35, University of Washington, Seattle WA 98195.
Supported in part by National Institutes of Health grant number LM04174 from the National Library of Medicine

1 Introduction

When developing new systems, it is sometimes necessary to incorporate subsystems that were developed by others. In modern process control systems, such subsystems often include an embedded computer that runs customized software. The documentation available for these pre-existing computer-based subsystems often fails to include a functional specification which is sufficiently detailed and explicit to support the design and analyses required for the new system. The original subsystem developers may be unavailable, or may be unable to provide information in a form that is consistent with the needs of the chosen software development method for the new system. Therefore, developers must infer a functional specification for the subsystem and then perform tests to confirm that the specification is sufficiently accurate for their purposes. This report describes our solution to one such problem that arose in a medical process control application. We have made certain simplifications here for purposes of exposition; the complete report appears in [1].

Our application is a large computer control system for a cyclotron and treatment apparatus used to deliver neutron radiation therapy for cancer. The facility has been in routine clinical operation for several years, using a control system provided by the cyclotron vendor [11, 10]. The University of Washington is now developing a successor control system [6, 5] to make the system easier and quicker to use and to support more complex treatment techniques.

We are attempting to achieve high reliability and safety at our facility by applying rigorous software development and quality assurance practices. A central idea in most recommendations and standards regarding software development is that developers should provide a functional specification that allows system outputs to be predicted if system inputs are described. We plan to incorporate several subsystems from the existing control system into our new control system. The existing documentation for these subsystems does not include functional specifications which are sufficiently detailed to permit their outputs to be calculated from their inputs in all situations of interest. Therefore we had to create the specifications and perform tests to confirm that that they are sufficiently accurate.

We decided to produce *formal specifications* in which the subsystems are described using some suitable mathematical notations. This will make it possible to calculate or infer system properties such as safety, using formal mathematical techniques.

We chose Petri nets [9, 8] as the formal specification notation for the subsystem described in this study. We have found Petri nets well-suited to our application, in part because they are easy for physicists and engineers to review [5, 1]. We also plan experiments with safety analyses based on Petri nets as described in [7], but those are not the subject of the present study.

We systematically derived the tests from the Petri net specification. Despite the huge literature on Petri nets, we are unaware of any reports in which Petri nets were used to derive tests. Our approach was suggested by methods for deriving tests from finite state machine models [3, 2].

2 System description

In this section we informally describe the control system that was the subject of this study.

Our facility is larger and more complex than most commercial therapy accelerators. The cyclotron produces a proton beam which travels through beam lines into either of two treatment rooms. Each treatment room includes a treatment head where the proton beam is used to produce a therapeutic neutron beam that irradiates the patient. The entire neutron therapy control system comprises a *proton beam control system* which controls the cyclotron and beam lines and two *treatment control systems* for the apparatus in the treatment rooms. Each of these three systems has its own control console and requires its own operator. Each can be further divided into subsystems.

This study concentrates on one of the treatment control subsystems, the *Dose Monitor Controller* (DMC). The DMC measures the dose emerging from the treatment head and is responsible for shutting off the beam when the prescribed dose has been delivered to the patient. It is the most safety-critical programmable element in the system because failures of the DMC could result in incorrect doses being delivered to patients.

Figure 1 illustrates the DMC and some other components of the treatment control system. Two *ion chambers* (IC1 and IC2) provide analog inputs to the DMC. The neutron beam emerging from the treatment head passes through both ion chambers and partially ionizes the air within. Each ion chamber is subjected to a DC voltage and the resulting current is proportional to the number of ionized air molecules and thus to the instantaneous radiation dose rate. Each ion chamber is connected to a *dose terminator* (DT1 and DT2) within the DMC. Each dose terminator includes a current-to-frequency converter, a digital counter, a preset dose register, a digital comparator, and an interlock relay. The current-to-frequency converter translates the ion chamber current into a train of digital pulses whose frequency is proportional to dose rate. The pulses are applied to the counter, which constitutes a digital integrator: the value of the count is proportional to the integrated dose delivered since the counter was last reset. The preset dose register holds a value proportional to the prescribed dose. The digital comparator opens the interlock relay when the value in the counter exceeds the value in the preset dose register, indicating that the prescribed dose has been delivered. This is the usual way to turn off the beam at the end of a treatment. The interlock relays in both dose terminators must be closed to enable the beam to be on.

There are two independent dose monitoring channels to guard against the possibility that one might fail.

The dose terminators are largely constructed of discrete electronic components, but they are controlled by a microcomputer embedded in the DMC. The microcomputer performs extensive monitoring and self-tests of the DMC analog and digital electronics, resets the counters, loads the preset dose registers, and performs some other functions. Each of these actions occurs at the appropriate time in the treatment sequence, which is ultimately under the control of the therapy operator. The operator's actions are mediated by another computer, called the *Control Computer* in Fig. 1.

The control computer and the DMC communicate by sending strings of ASCII text back and forth over an RS-232 serial line. For example, before each treatment begins, the control computer sends a command to the DMC to load the preset dose registers with the prescribed dose. The DMC responds to each command by sending an acknowledgement string or an error message. The DMC can also send unsolicited messages to the control computer, for example when a dose terminator opens its interlock relay at the end of a treatment.

There is a third interlock relay in the DMC called the *timer relay*. This relay must also be closed to enable the beam to turn on. A "watchdog timer" circuit ensures that this relay can be closed only when the microcomputer provides repetitive pulses at a high rate under program control. The microcomputer program generates these pulses only during those portions of the treatment sequence whenthe beam is allowed to be on. To provide an additional safety check in case both dose terminators should fail, there is an elapsed time counter which is implemented by the microcomputer program and an external counter/timer circuit. A preset time register is loaded with the estimated duration of each treatment, calculated by dividing the prescribed dose by the nominal dose rate of the machine. The microcomputer causes the timer interlock relay to open when the value in the elapsed time counter exceeds the value in the preset time register, indicating that the estimated treatment time has passed. This turns of the beam, if it is not already off. A typical treatment lasts about one or two minutes.

Another input to the DMC comes from *Faraday Cup 1* (FC1). The Faraday Cup is a barrier that can be dropped into the beam line to prevent the beam from reaching either treatment room. This information is useful to the DMC because the neutron beam cannot possibly be on in a treatment room when this Faraday Cup is closed.

The values of the preset time, preset dose, elapsed time, and measured dose (in both channels) held in the DMC registers and counters are continuously shown on dedicated numeric displays on the treatment operator's console.

In the present system configuration, the three DMC interlock relays control a single relay

(Insert Figure 1.1 from Alex Barke thesis)

Figure 1: Dose Monitor Controller (DMC) and other treatment control components

called the *dosimetry relay* in another treatment subsystem called the *hard-wired safety interlock system*, or HSIS (see Fig. 1). When any switch in the HSIS is open, the beam is off and cannot be turned on.

The position of Faraday Cup 1, and the state the dosimetry relay in the HSIS, are displayed at the cyclotron control console. The state of the three interlock relays internal to the DMC are not normally displayed and were not monitored in this study.

When all of the switches in the HSIS are closed, an operator may turn on the beam by pressing a button on the console. While the beam is on, the DMC performs additional control functions besides those required to shut the beam off when the treatment is complete. Based on the instantaneous dose rate, it generates correction signals that are used to keep the dose rate nearly constant and to steer the beam. The mechanism for turning on the beam, and the analog control functions performed by the DMC, were not the subject of this study and will not be explained further.

The microcomputer in the DMC is based on a Z80 microprocessor. Its control programs are written in assembly language and the object code occupies about 13 kilobytes in read-only memory (ROM). Most of the code appears to deal with the monitoring, self-test and analog control functions, not with the beam termination functions that were the subject of this study.

We have available block diagrams and detailed circuit diagrams for the DMC hardware, and flow charts and program listings (in assembly language) for its control programs. Some of these sources provide helpful background information, but we did not attempt to infer detailed functional properties of the DMC from them. We did not have any information regarding earlier testing of the DMC.

3 Problem statement

We plan to retain the DMC in the new system. The DMC is a self-contained unit which provides needed functions and has performed well. It includes complex analog and digital electronics which are beyond the scope of our project to duplicate.

The DMC has performed safely for several years in the present system configuration, but this does not provide assurance for the new system. In the present system, the control computer always sends a limited set of commands in a nearly fixed order with similar time intervals between commands, as determined by the particular treatment sequence that the control computer software now provides. An important motivation for developing the new control system is to support more flexible and efficient treatment sequences, possibly

involving different DMC command sequences and less time between commands. We were concerned that some different sequence of commands (that does not occur in the present system) might cause the dosimetry relay to close at the wrong step in a treatment sequence, or cause the DMC to halt or "hang" while the beam was on, leaving the dosimetry relay closed. Such events might permit the beam to be turned on at an inappropriate time, or to remain on indefinitely.

We limited the scope of this investigation to the beam-termination functions of the DMC. We did not consider monitoring, self-test and analog control functions. Testing those functions would require partially disassembling the DMC and injecting erroneous signals, which was not practical to do at this time.

3.1 Available documentation

The primary documentation we had available was a brief description, in English, describing the protocol and commands used to communicate with the DMC over the serial communications line. We initially identified eight commands as relevant to this study, whose (paraphrased) descriptions are shown in Table 1. The description also includes a sample listing of the messages exchanged to accomplish a typical treatment.

The documentation included descriptions of error conditions. The five errors considered in this study are shown in Table 2. The documentation also says that if an error occurs while a treatment is in progress, the treatment will be stopped.

We performed some preliminary experiments to check the accuracy of the documentation. We connected a printer to make a permanent record of the message traffic in both directions between the DMC and the control computer during several actual treatments. Normally this traffic is not visible to the operators, and is not recorded. In order to confirm the error codes, we disconnected the DMC from the control computer and attached a terminal. We typed commands at the terminal keyboard and observed the acknowledgement strings and error messages on the terminal display.

3.2 Problems with the documentation

We discovered several problems with the documentation:

The messages we observed during actual treatments were not exactly the same as in the "typical" example shown in the documentation. In particular, a different sequence of initialization commands for the analog control functions appeared. Moreover, STOP and

RESET	Reset DMC and initialize some parameters with default values.
SELECT	Perform self-test and initialize console displays. This command must be given prior to every treatment.
SET-TIME	Load treatment time.
SET-DOSE	Load prescribed dose.
START	Close interlock relays, making it possible to begin a treatment. This command must be preceded by SET-TIME and SET-DOSE. This command will result in FC1 opening. When FC1 opens, the elapsed timer begins counting.
STOP	Interrupt a treatment so it can subsequently be continued. The treatment may be continued without the usual start up procedure. Opens an interlock relay, which causes FC1 to close. The elapsed timer stops.
CONTINUE	Continue a treatment after STOP. Closes the HSIS, which opens FC1. The elapsed timer continues.
TERMINATE	Perform self-test at end of treatment. This is different from the self-test performed by SELECT.
TEST	Enable analog functions to be tested. Tests may be performed without the usual start up procedure. WARNING: This command sets preset dose and time to large values. This command opens FC1.

Table 1: DMC commands

ERROR 01	Command string cannot be interpreted.
ERROR 30	START attempted before SELECT performed.
ERROR 31	START attempted before SET-DOSE performed.
ERROR 32	START attempted before SET-TIME performed.
ERROR 33	SET-TIME or SET-DOSE attempted while treatment in progress.

Table 2: DMC error codes

CONTINUE appeared in the example but we never observed these during actual use of the system. This revealed that the documentation was not always clear regarding which commands were optional and which were required in particular situations.

The documentation implies constraints on the ordering of some commands, but appears not to describe the constraints completely. Some of the error codes indicate an incorrect sequence of commands, but other seemingly incorrect sequences have no corresponding error codes. For example, there is no error code for "CONTINUE attempted before STOP performed." The effect of attempting such sequences was not clear.

The documentation implies that both the CONTINUE and TEST commands can close the dosimetry relay, without going through some of the steps usually required at the beginning of a treatment. However the documentation was not clear about what conditions did have to be satisfied in order for these potentially hazardous commands to be performed.

The documentation is unclear regarding direct and indirect effects. Several commands are said to cause Faraday Cup 1 (FC1) to open or close. In fact there is no output from the DMC directly to FC1. Instead, FC1 is interlocked with the HSIS, so that if any switch in the HSIS opens for any reason, then FC1 closes. The dosimetry relay controlled by the DMC is just one of the switches in the HSIS that can cause this to happen. When the HSIS closes again, FC1 may open under operator or program control (but it need not).

Due to confusion arising from the language about FC1, it is not clear whether the elapsed timer is (in effect) gated by the FC1 input, or whether it is simply turned on and off by the DMC program as the dosimetry relay is opened and closed, regardless of the state of FC1. This could become important if the policy linking FC1 and HSIS changes in the new system. Also, a comment about FC1 seems to imply that the TEST command enables the beam to come on, which appears potentionally hazardous.

We also observed a potential problem when we were investigating the error codes. The first

time an erroneous command was presented, the DMC responded with the appropriate error message. However when the erroneous command was simply repeated, the DMC responded with the usual acknowledgement string, not an error message. We became concerned that successive errors might not be handled safely.

4 Method

We proposed an initial Petri net model, designed tests intended to determine whether the model was accurate, performed the tests, and created a final model that was consistent with the results of the tests.

4.1 Initial Petri net model

We proposed an initial Petri net model of the DMC, based on our understanding of the documentation and our experience using the existing system. We represented the DMC, along with certain beamline components including FC1, as a collection of concurrent processes. Each process is represented as a cycle composed of alternating conditions and transitions. The system usually traverses one circuit of each cycle for each treatment. The beginning of each cycle is taken to be the condition that exists when the therapy operator is preparing to begin the treatment. Interactions between processes are represented as transitions that belong to more than one cycle.

For example, figure 2-A shows our initial model for one dose terminator. Transitions represent receipt of the commands SET-DOSE and TERMINATE. Transitions also represent transient events: at the transition labelled "*", the value of the dose counter exceeds the preset dose. Figure 2-B shows our initial model for the dosimetry relay. One transition represents the START command; the transition labelled "*" occurs when the value of one of the dose counters exceeds the preset dose. In fact the two transitions labelled "*" in Fig. 2 represent the *same* event, and the two nets could be redrawn (less neatly) as one single net.

Figure 3 shows our initial model, obtained by merging the nets shown in Fig. 2 with several others. (This figure is taken from [1] and uses slightly different naming conventions, as well as several nonstandard notational conveniences in an attempt to reduce clutter). We did not attempt to represent erroneous command sequences in this initial model, because the documentation did not provide sufficient guidance.

The entire initial model was drawn by hand and inspected by eye. We executed the model

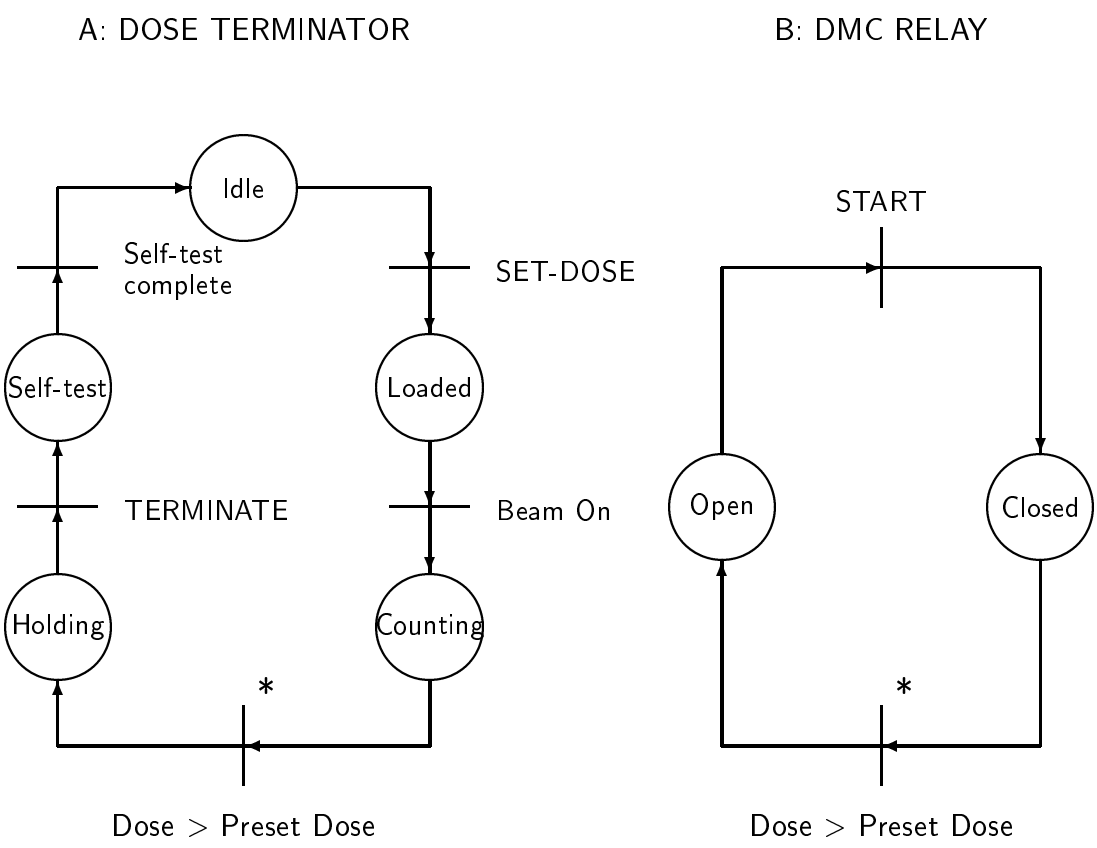

Figure 2: Petri nets for dose terminator and dosimetry relay

(Insert Fig 4-3, p. 33 from Alex Barke thesis)

Figure 3: Initial Petri Net model of DMC and associated components

by hand to confirm that several typical treatment sequences were correctly represented. No automatic syntax checking or model execution was performed.

4.2 Designing the tests

From the initial Petri net model, we derived a series of tests intended to reveal whether our provisional specification accurately characterized controller behavior. The tests were designed to systematically traverse every arc of the Petri net comprising the initial model and also to present errors that should be handled in a reasonable way.

Each test was based on a sequence of transitions and conditions found in the initial model. After choosing the sequence, a test script was written, listing the actions that the operator must perform in order to force the transitions, and the observations the operator should make to confirm that the expected conditions occurred. Usually these instructions involved pressing buttons on the consoles and observing the displays.

The sequence that formed the basis for each test was derived by inspecting the hand-drawn model, and the operator's actions and observations were determined based on our understanding of the consoles.

We planned three series of tests: expected conditions with expected transitions, expected conditions with erroneous transitions, erroneous conditions with expected transitions.

4.2.1 Expected conditions with expected transitions

The first series of tests examines the cases where we expected no errors to be detected by the DMC. These tests were designed by finding several sequences of transitions and conditions which, together, ensured that every arc in the combined initial model was traversed at least once, and which were possible to perform using the controls usually available to operators.

Five such sequences were required in order to cover all arcs, each corresponding to a particular treatment scenario: a treatment that is stopped by one of the dose terminators, a treatment that is stopped by the timer, and three kinds of interrupted treatments including use of the STOP and CONTINUE commands.

4.2.2 Expected conditions with erroneous transitions

The second series of tests was designed to investigate error handling by the DMC command interpreter, by presenting commands at the "wrong" step in the treatment sequence ("wrong" according to our initial model). From each test sequence in the first series, many new test sequences were derived. In each new sequence, the operator proceeds through a certain number of steps exactly as before, but upon reaching a transition which corresponds to receipt of a command, causes the DMC to receive a different command than would be expected (based on the initial model). For each such transition, there were tests that attempted all of the commands considered in this study (RESET, SELECT, SET TIME, SET-DOSE, START, STOP, CONTINUE, TERMINATE and TEST) as well as a nonsense string which did not belong to the command set. Successively longer scripts proceeded further through the sequence before presenting an unexpected command. After the unexpected command, the script called for the operator to attempt the command a second time and then to attempt to continue with the normal sequence.

4.2.3 Erroneous conditions with expected transitions

The third series of tests was designed to investigate how the DMC handled unexpected combinations of conditions. These tests were created by inspecting the initial model and finding transitions which required more than one enabling condition. We tried to create a test script which would create a similar situation in which one or more of the enabling conditions were not met, but the expected transition was attempted (usually by sending some command).

Many of these tests called for sending a command ahead of several commands that precede it in the normal sequence; of these, some were identical to tests in the second series and were not repeated. Other tests involved unusual use of Faraday Cup 1. Some situations in this category could not be scripted because they would be infeasible to perform.

4.3 Performing the tests

We performed the tests in a realistic context, with the entire control system intact (not just the DMC), the cyclotron running and the beam on in the treatment room (with no patient present). This was necessary in order to drive the dose terminators with real signals. The usual cyclotron and therapy operators followed the test scripts while the testers recorded observations. A printer was connected to record all traffic between the DMC and the control computer. The following sections describe special interventions that had to be made for

particular tests.

4.3.1 Expected conditions with expected transitions

These tests were performed by the cyclotron and therapy operators following the usual treament sequence. In some tests the cyclotron operator had to manually reduce the dose rate of the machine to ensure that the timer, not a dose terminator, stopped the treatment.

4.3.2 Expected conditions with erroneous transitions

To perform these tests, it was necessary to disconnect the control computer from the DMC communication line. Instead, a terminal was connected and one of the testers typed the commands required by the test script.

4.3.3 Erroneous conditions with expected transitions

These tests also required the DMC to be driven by a tester typing at a terminal. Some tests required the cyclotron operator to control the beam and Faraday Cup, using a "test mode" in which the HSIS is reconfigured to remove some of the interlocks that are active during patient treatments. In a few tests a cyclotron engineer disconnected Faraday Cup 1 from its control system output and drove it under manual control as required by the test script.

4.4 The final model

We constructed a final model, based on the test results. We executed the model by hand to confirm that it was consistent with observations made during all the tests.

5 Results

Each series of tests revealed discrepancies between observed behavior and our initial model. The following sections describe some of the observations that informed the final model.

5.1 Expected conditions with expected transitions

The initial model adequately describes normal treatments that are terminated by a dose terminator or the timer.

The scripted operator actions that were supposed to invoke the STOP and CONTINUE commands failed to do so. We had to force these commands by using a special diagnostic program. This is apparently the only way that the present control system allows these commands to be used.

A test that used the STOP and CONTINUE commands revealed one minor error in the initial model, concerning the conditions that exist after the self-test performd by the TERMINATE command.

5.2 Expected conditions with erroneous transitions

RESET can be performed without error anytime; it always opens the dosimetry relay if it was closed.

One of the analog control commands, SET-DOSERATE, has to be sent between RESET and START. This command had to be added to all the test scripts in order to proceed with the study.

The DMC command interpreter enters the its Reset condition after performing the TERMINATE command, and after many (but not all) errors. In this condition, the dosimetry relay is open, and many commands are not considered erroneous. That explains why we sometimes see no error message when an erroneous command is repeated.

After RESET and before START, SET-DOSE and SET-TIME may be sent in any order, and may be repeated indefinitely many times. Successively SET values simply overwrite the previous values. Futhermore, after SELECT is sent, then prior to START, either SET- as well as SELECT may be repeatedly sent in any order.

Error-handling is more robust than described in the original documentation. Most commands elicit Error code 33 if they are sent after START. The documentation (Table 2) says that this error means, "SET TIME or SET DOSE attempted while treatment in progress." However, the text of the error message that actually appears (on the RS-232 line back to the control computer) is "Error 33; command not allowed!" When this message appears, the dosimetry relay opens and the command interpreter enters its Reset condition. After START, the commands that result in this message are SET-TIME, SET-DOSE SELECT,

START, CONTINUE, TERMINATE, and TEST. Similarly, after STOP the commands SET-TIME, SET-DOSE, SELECT, START, and TEST elicit Error 33 and the Reset condition. And likewise, after a dose terminator or the elapsed timer stops the treatment, but before TERMINATE is sent, then SET-TIME, SET-DOSE, SELECT, START, and TEST elicit Error 33 and the Reset condition.

A nonsense string always elicits the error message, "Error 01; Syntax error!". After START it opens the dosimetry relay and proceeds to the Reset condition, otherwise there is no further effect.

STOP and CONTINUE are usually accepted without invoking an error message. STOP has no effect unless it is sent after START. CONTINUE does not cause the dosimetry relay to close unless it follows STOP.

TEST does not close the dosimetry relay. It appears to be an alternative to a series SET-commands.

After the beam turns off, or after STOP, TERMINATE invokes the final self-test, succeeded by the Reset condition. After START, TERMINATE elicits Error 33 and the Reset condition. In other conditions, TERMINATE causes the DMC to enter a "hung" state in which it stops responding to other commands until RESET is sent.

5.2.1 Erroneous conditions with expected transitions

In the Reset condition, the elapsed timer does not count up even if Faraday Cup 1 is open. After START, the elapsed timer counts when FC1 is open and stops when FC1 is closed. If FC1 is prevented from closing after STOP (this does not normally occur), the elapsed timer continues counting.

6 Conclusions

This study did not reveal any serious problems that indicate the DMC should not be included in the new system. We examined the DMC over a greater range of situations than previously observed, but did not find any command sequences that would enable the beam to turn on at the wrong time, or remain on indefinitely. We found that the DMC has better error checking than a skeptical interpretation of the documentation might have implied.

We conclude that building a formal model for an existing subsystem that must be incorpo-

rated into a new system is helpful. The exercise of building the model is useful for organizing observations and planning tests, and encourages consideration of contingencies that might otherwise be overlooked.

7 Discussion

Any testing is necessarily incomplete and inconclusive. We realize that some combination of circumstances that we did not examine in this study might cause the DMC to fail in a potentially hazardous fashion. We did not systematically vary the timing between commands; commands were simply presented at convenient speed. We did not repeat the same test sequences multiple times, to simulate what might occur during a day of successive treatments; we assumed that the DMC has no important "memory" that persists across RESET transitions. We did not examine the effect of varying the preset time or prescribed dose; we assumed that the DMC handles all values of these variable parameters in the same way. We did not consider any interactions between the beam termination functions and the other monitoring, self-test and analog control functions of the DMC. In particular, spontaneous errors resulting from failed self-tests were not examined. These should occur, for example, when the doses measured by the two dose terminators differ significantly.

Our particular choice of formal notation is not central; it seems likely that many of the benefits we derived from using a formal model would also have occured if we had used a different notation than Petri nets. We chose Petri nets because we are familiar with them. However, Fig. 3 illustrates the difficulty of representing Petri nets of useful size. The worst difficulties arise when it is necessary to model transitions that are allowed from many different conditions, such as those caused by our RESET command. These can be represented, but the drawings become cluttered. These situations are more neatly represented in other graphic notations, such as Statecharts [4].

8 Acknowledgements

The authors thank Lee Hutter, Sandra Poirier, Astor Rask, and Jerry Sintay for their assistance performing the tests, and Ira Kalet and Alan Shaw for their comments on the study.

References

[1] Alexandra Barke. Use of Petri nets to model and test a control system. Master's thesis, University of Washington, Seattle, Washington, 1990.

[2] Jonathan A. Bauer and Alan B. Finger. Test plan generation using formal grammars. In *Proceedings of the Fourth Annual International Conference on Software Engineering*, pages 425–432. IEEE, 1979.

[3] T. S. Chow. Testing software design modeled by finite state machines. *IEEE Transactions on Software Engineering*, SE-4:178 – 186, 1978.

[4] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[5] Jonathan Jacky. Formal specifications for a clinical cyclotron control system. In Mark Moriconi, editor, *Proceedings of the ACM SIGSOFT International Workshop on Formal Methods in Software Development*, pages 45–54, Napa, California, USA, May 9–11 1990. (Also in *ACM Software Engineering Notes*, 15(4), Sept. 1990).

[6] Jonathan Jacky, Ruedi Risler, Ira Kalet, and Peter Wootton. Clinical neutron therapy system, control system specification, Part I: System overview and hardware organization. Technical Report 90-12-01, Radiation Oncology Department, University of Washington, Seattle, WA, December 1990.

[7] Nancy G. Leveson and Janice L. Stolzy. Safety analysis using Petri nets. *IEEE Transactions on Software Engineering*, SE-13(3):386–397, 1987.

[8] Tadao Murata. Petri nets: Properties, analysis, and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[9] James L. Peterson. Petri nets. *ACM Computing Surveys*, 9(3):223–252, 1977.

[10] R. Risler, S. Brossard, J. Eenmaa, J. Jacky, I. Kalet, A. Rask, and P. Wootton. Routine operation of the Seattle cyclotron facility. In *Proceedings of the Twelfth International Conference on Cyclotrons and Their Applications*, 1989.

[11] Ruedi Risler, Jüri Eenmaa, Jonathan P. Jacky, Ira J. Kalet, Peter Wootton, and S. Lindbaeck. Installation of the cyclotron based clinical neutron therapy system in Seattle. In *Proceedings of the Tenth International Conference on Cyclotrons and their Applications*, pages 428–430, East Lansing, Michigan, May 1984. IEEE.