Control System Specification for a Cyclotron and Neutron Therapy Facility

Jonathan Jacky, Ruedi Risler, Ira Kalet, Peter Wootton,
Alexandra Barke*, Stan Brossard, and Ralph Jackson

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

Abstract

It is usually considered an essential element of good practice in engineering to produce a specification for a system before building it. However, it has been found to be quite difficult to produce useful specifications of large software systems. We have nearly completed a comprehensive specification for the computer control system of a cyclotron and treatment facility that provides particle beams for cancer treatments with fast neutrons, production of medical isotopes, and physics experiments. We describe the control system as thoroughly as is practical using standard technical English, supplemented by tables, diagrams, and some algebraic equations. This specification comprises over 300 single-spaced pages. A more precise and compact specification might be achieved by making greater use of formal mathematical notations instead of English. We have begun work on a formal specification of our system, using the Z and Petri net notations.

1 Introduction

The Clinical Neutron Therapy System (CNTS) at the University of Washington is a cyclotron and treatment facility that provides particle beams for cancer treatments with fast neutrons, production of medical isotopes, and physics experiments [12,11]. The facility was installed in 1984, and includes a computer control system provided by the cyclotron vendor. Devices under computer control include a 900 amp electromagnet and a 30 ton rotating gantry, as well as four terminals at three operator consoles. The control system handles over one thousand input and output signals, and includes six programmable processors as well as some nonprogrammable (hard-wired) controls

The University is now developing a new, successor control system. This development project is motivated by requirements to make the system easier and quicker to use, easier to maintain, and able to accomodate future hardware and software modifications.

*Partially supported by National Institutes of Health grant number LM04174 from the National Library of Medicine

†Partially supported by NIH contract no. CM97282 from the National Cancer Institute

We have high reliability and safety requirements. There is growing recognition that the development of software which controls medical devices is not sufficiently systematic, that this is adversely affecting costs and safety, and there is much room for improvement [6].

We are attempting to achieve high reliability and safety by applying rigorous software development and quality assurance practices. We determined that our first step in this project should be the production of a comprehensive specification for the new control system, to serve as an authoritative and complete guide for software development, testing, and instruction of facility users.

2 Why write a specification?

It is usually considered an essential element of good practice in engineering to produce a specification for a system before building it. Modern recommendations for the development of safety-critical computer-based systems [2,8] emphasize the need for an explicit statement of functional and safety requirements, and a development process that can be shown to produce an implementation that meets the requirements. A central idea is that developers should provide a functional specification that allows system outputs to be predicted if system inputs are described.

However, it has been found to be quite difficult to create useful specifications for large software systems. A computer scientist has articulated the widely-held view that it is not practical to produce them:

> For many systems, it is simply not possible to construct a compact, concise specification. There are many different kinds of data and conditions, and a complete specification would have to detail the actions of the system for all of them. In practice such a specification is just a program. The working out of the many details only occurs as the system is being programmed. This has been observed to be a common feature of real-time and data processing systems [4].

Although we appreciate the difficulties, we do not agree with this view. Our experience reviewing the program code

0-7803-0135-8/91$03.00 ©IEEE

for the existing control system has made it clear that programs are no substitute for specifications.

3 Why programs are no substitute for specifications

We have been asked, "Why do you need to write a specification, when the program code that you write will tell you everything that the system will do?" We have found this attitude to be so widely held that it requires a detailed answer.

An obvious answer concerns project scheduling. We would like to have the acceptance test procedure and the instruction manuals for the users ready as soon as the programming is finished. To enable the acceptance tests to be designed and the manuals to be prepared while programming is underway, it is necessary to have a description of what the program will do, which is available before the program is complete. If this information has to be extracted from the program after it is written, testing and user instruction must be delayed.

Another answer is that there needs to be an independent standard of accuracy that the program can be tested against. This standard should be set by the users; in our case these are the physicists, engineers, and therapy technologists who use and maintain the facility. These are not the same people who will write the programs, and it is not reasonable for them to review program code to determine whether their needs will be met.

An answer that is less obvious to many software developers is that program code is not a suitable medium for expressing the information that needs to be in a specification. This is not merely a matter of the degree of expressivity provided by different notations. In fact, it is generally impossible to invert program code to determine the requirements that it is supposed to satisfy. When a program is written, requirements information is lost, but a great deal of information must be added that is not determined by the requirements, but is needed to code the program in the chosen programming language and get it to run on the available facilities.

4 Our specification

We have nearly completed a specification for the new control system. We believe it is as comprehensive as is practical to express using standard technical English, supplemented by tables, diagrams, and some algebraic equations.

The specification consists of four parts. Part I is an overview of the system that describes the facility, the hardware organization of controls, and introduces much of the vocabulary used in subsequent parts. It comprises almost 100 pages of single-spaced text. Part II is a detailed specification of operations which users perform at video terminals and control consoles. It comprises about 150 pages, including many illustrations of displays. Part III will be a detailed specifications of internal operations involving the cyclotron and therapy apparatus itself, which are only indirectly visible to users, and will comprise about 100 pages. the Part IV will be a largely tabular presentation of numerical operating parameters whose values can be changed independently of the rest of the specification. We plan to maintain these tables in a relational database.

Part I has been prepared for distribution outside our department as a technical report [7] and we anticipate releasing the other parts as they are completed in coming months.

5 How the specification was written

The specification is largely based on information gathered during meetings with facility users. The code and other documentation for the existing control system provide some useful information but have not been nearly as important as the interviews. The documentation for the existing system does not include a specification in the sense used here.

Most of the information was gathered in meetings that included the first two authors, a computer specialist (JJ) and the chief engineer of the facility (RR). Some meetings included other participants as well and there were meetings that did not include these authors. Most meetings lasted 2 to 3 hours. Computer specialist(s) asked questions; facility user(s) answered while a computer specialist took notes.

The first author drafted each chapter of the specification based on information gathered in a series of meetings, usually working from his own notes, sometimes from written material contributed by others. The chief engineer reviewed the chapter and the two met again. The review and ensuing discussion invariably revealed important errors, omissions and new information.

The first author then prepared a second draft. Second drafts were circulated among all authors for review. This review always resulted in some substantive revisions and many stylistic ones.

The first author then prepared the final version.

We believe that this lengthy process is the minimum necessary to achieve an acceptable specification because significant changes were always made after each cycle of review. In some cases more than two revisions were necessary.

The speed of the process was limited by the amount of time that facility users could spare from their other duties to meet with computer specialists and review drafts.

Most effort was devoted to creating and reviewing written material, but we also did some programming to experiment with the appearance of video screen displays and other aspects of the human-computer interface.

6 Formal specifications

Our specification is written in standard technical English, supplemented by tables, diagrams, and some algebraic equations. It is organized to help facility users read and understand it, so they can offer meaningful reviews. However, this organization is not necessarily the best for purposes of programming and analysis. An alternative is to make greater use of mathematical notations, annotated sparingly with English. Such specifications are called *formal specifications.*

We believe that there may be advantages to developing programs from formal specifications. They can be more precise and compact than English specifications. The can be checked by machine for certain kinds of errors. They can be used to prove or calculate whether the specificied behavior is consistent with certain intended properties, such as safety.

Formal specifications are distantly related to popular design notations based on data flow diagrams, which have been applied to acclerators [10]. We are investigating notations that provide more mathematical rigor, in the sense that they allow more properties to be inferred or calculated. Our preliminary experiments using the Petri net notation [9] are reported in [1], and some investigations into the Z notation [3,13] appear in [5].

7 Further work

We are preparing to implement the system we have specified. We will determine whether our specifications actually do provide sufficient information to build a useful system, and will assess the usefulness of formal specifications.

8 Acknowledgements

The authors thank Lee Hutter, Sandra Poirier, Astor Rask, and Jerry Sintay for sharing their knowledge of the facility.

References

[1] Alexandra Barke. *Use of Petri Nets to Model and Test a Control System.* Master's thesis, Department of Computer Science and Engineering, University of Washington, Seattle, Washington, 98195, 1990.

[2] Great Britain Health and Safety Executive. *Programmable Electronic Systems in Safety Related Applications. Volume 1: An Introductory Guide. Volume 2: General Technical Guidelines.* Her Majesty's Stationery Office, London, 1987.

[3] Ian Hayes, editor. *Specification Case Studies.* Prentice Hall International, Englewood Ciffs, NJ, 1987.

[4] William E. Howden. Validating programs without specifications. In Richard A. Kemmerer, editor, *Proceedings of the ACM SIGSOFT '89 Third Symposium on Software Testing, Analysis and Verification (TAV3)*, pages 2 – 9, ACM Press, 1989. (Also published as ACM SOFTWARE ENGINEERING NOTES, 14(8), Dec. 1989).

[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. Programmed for disaster: software errors that imperil lives. *The Sciences*, 29(5):22–27, 1989. September/October.

[7] 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.

[8] Nancy G. Leveson. Software safety: what, why and how. *ACM Computing Surveys*, 18(2):125–163, June 1986.

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

[10] G. A. Ludgate, B. Haley, L. Lee, and Y. N. Miles. The use of structured analysis and design in the engineering of the TRIUMF data acquisition and analysis system. *IEEE Transactions on Nuclear Science*, NS34(1):157 – 161, 1987.

[11] R. Risler, S. Brossard, J. Eenmaa, J. Jacky, I. Kalet, A. Rask, and P. Wootton. Routine operation of the Seattle cyclotron facility. In B. Martin and K. Ziegler, editors, *Cyclotrons and Their Applications: Proceedings of the Twelfth International Conference, Berlin, FR Germany, 8 – 12 May 1989*, World Scientific Publishing Co., Singapore, 1991.

[12] 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, IEEE, East Lansing, Michigan, May 1984.

[13] J. M. Spivey. *The Z Notation: A Reference Manual.* Prentice-Hall, New York, 1989.