# A SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT MODEL FOR THE MEDICAL DEVICE INDUSTRY

Janis V. Halvorsen
Food and Drug Administration
60 Eighth Street N.E.
Atlanta, GA 30309

Soheil Khajenoori
Dept. of Electrical & Computer Engineering
University of Central Florida
P.O. Box 25000
Orlando, FL 32816-0993

## ABSTRACT

Medical devices that were entirely controlled by hardware systems are being replaced with software controlled systems. The versatility of software and ease of change means that documenting that software should be given a very high priority with designers, developers, and users of these software based systems. The Food and Drug Administration (FDA) regulates medical devices. Medical devices can contain software or be stand alone software. FDA's study of software related recalls from 1983 to 1989 has found that the number of recalls have quadrupled over that time period. Eighty nine per cent of the recalls were specification related. The software requirements specification document defines the what of the system and is used to design, test, and validate software. A model of a requirements specification document for use by the medical device industry is discussed. The model is not endorsed by the Food and Drug Administration nor proposed as the only model that could be used to meet the regulatory requirements. The model is based on the ANSI/IEEE and Fairley models. The model can also be used to evaluate preexistent software requirements specification documents.

## INTRODUCTION

Medical devices, controlled entirely by hardware systems, are being replaced with software controlled systems. The versatility of software and convenience of change means that documentation should be given a very high priority by designers, developers, and users of software based systems. Unfortunately, instead of documentation being part of the initial design, it is often left as an afterthought. Designers of hardware do not sit down with wires, op amps, capacitors, etc. and put a device together. Documentation has always existed for hardware controlled medical devices. Hardware has had its requirements, specifications, blueprints, maintenance documents, testing setups, and procedure manuals for disassembly, trouble shooting, reassembly, etc.

Why then should software be any different? Often comments in the code are the only documentation. Software code for any computer system needs to have additional accompanying documentation that has been developed prior to the first line of code being written. This initial documentation needs to be dynamic, useable, and changeable. Its beginning should coincide with the initial discussions of the system requirements and should be functional when the software is in the maintenance phase of the software life cycle.

Many levels of documentation are necessary to adequately document software that is or is a component of a medical device. One of the first documents is the software requirements specification. Specifications for medical devices can come from many sources. A company may decide what a device will do and then attempt to market that device. A doctor may have a device that does much of what he wants but needs additional features. He may ask a company to manufacture a new device to his specifications. A company may have a device that was controlled electronically and now wants their new model to be computer controlled. The user is the person who is having the software developed for him. The user in each of these examples is typically the marketing department, the doctor, and the president of the company.

Documentation needs to be complete and detailed enough so it is fully understandable to a knowledgeable individual who is reviewing the documentation. Who that knowledgeable individual is would depend upon the document. For example, the software requirements specification document should be understandable to the user and to the developer of the requirements [12]. The user cannot abdicate his responsibilities of involvement in developing the software specification requirements. Even if the user contracts with a person or company who has designed similar software systems in the past, software can and will be customized. A user who purchases off the shelf software must develop requirements because he will need to assure that the software is adequate for his needs and meets all of his requirements. Communication is the primary function of the requirements specification.

The cost to correct a problem is very little during the software requirements specification phase when compared to the final testing and integration stage of the software life cycle (according to Good [5] in 1986 - $350 compared to $12,000). These problems can be avoided by starting with a good software requirements specification document [8, 11]. This paper focuses on that document. The software requirements specification document should be updated and current throughout the life cycle of the medical device. Software used in or as a component of a medical device must have complete and accurate documentation. A review of medical device recalls from 1983 to 1989 shows that the number of software related recalls has tripled; 89% of which were attributed to inadequate design [14].

## MEDICAL DEVICE SOFTWARE REGULATION

The U. S. Food and Drug Administration (FDA) is a governmental regulatory agency with the responsibility for assuring the safety and efficacy of medical devices. A medical device is defined in the Federal Food, Drug, and Cosmetic Act (FD&C Act) as "any instrument, apparatus, implement, in vitro reagent, or similar or related article ...which is (1) recognized in the official National Formulary, or the United States Pharmacopeia, or any supplement to them, (2) intended for use in a diagnosis of disease or other conditions, or in the cure, mitigation, treatment, or prevention of disease, in man or other animals, or (3) intended to affect the structure or any function of the body of man...and which does not achieve its principal intended purposes through chemical action within or on the body of man...and which is not dependent upon being metabolized for the achievement of any of its principal intended purposes" (21 USC 321[h], 1990 [18]) [2, 9]. Software that is a medical device includes data and/or programs that operate a computer controlled system; an expert system; or a database contained on disks, tapes, or embedded in the hardware of a device (also referred to as firmware). A software package that is a library function only, is not currently regulated as a medical device, if it is not used as the sole basis to make a diagnosis.

The FDA enforces the Good Manufacturing Practices (GMPs) for medical devices contained in Title 21 Code of Federal Regulations (CFR) parts 800 - 1299 [20]. This section of the regulations enumerates the documentation needed. Manufacturers are responsible for assuring that their devices comply with the GMPs. Software can be part of a computerized device, such as a computerized tomography device, or a stand alone package, such as software to set up a radiation therapy machine given a physician's prescription.

The GMP regulations are written as broad requirements setting forth standards and practices to be followed by manufacturers. The regulations must be interpreted for application to the particular device and manufacturing facility. Two basic premises of the GMP's are that 1) quality begins at the design phase; and 2) quality cannot be tested into a product. Since software design cannot be done properly without requirements, quality begins with the requirements specification phase. Without adequate documentation of the specification requirements, quality cannot be assured in the design of the final device.

After the device is ready for distribution, FDA requires all documents to be complete, reviewed, and approved by responsible and knowledgeable individuals within the company. The Safe Medical Device Act of 1990 amended the FD&C Act so FDA can promulgate regulations requiring manufacturers to do pre-

0-7803-1257-0/93/$3.00 © 1993 IEEE.

production design validation [19]. The design is better and validation is easier when the software requirements specification document is complete and accurate.

## SOFTWARE REQUIREMENTS SPECIFICATION

Develop documentation in a systematic way, following a plan already written (such as the IEEE Software Configuration Management Plan [16]) or developed in-house. The goal is to assure consistency of documentation. Written standards should include the qualifications of personnel involved, their training, documentation standards, approaches to design, and coding and testing standards. Written software documentation procedures should describe the documents to be used, who will maintain the documents, and where the documents will be located. Software configuration management is equivalent to the term 'change control' used by the GMPs. Its goal is to assure that only the current, tested, reviewed, and approved version of the software and software documentation is used.

R. E. Fairley [4] in Software Engineering Concepts and the ANSI/IEEE standard 830 - 1984 [15] define the characteristics and content of the software requirements specification document. R. Kosmala [10] divides the requirements specification into two documents: functional requirements and system specification. The software requirements specification document should contain all the requirements that the medical software needs to fulfill to meet specifications. The document should not be hardware specific. If hardware has already been developed or is pre-existent, the hardware will impose design constraints that should be included in the software requirements specification document. How the software is to be written should not be included since this is part of the design of the software.

The software requirements specification defines the "what" of the system and not the "how". Write in a way to make it adaptable to other hardware systems or software languages. It should be unambiguous, consistent, verifiable, modifiable, comprehensive, complete, and easy to use at all times. To be unambiguous, the language must be precise and consistent throughout the document. To be verifiable, define the requirements in quantifiable or testable terminology. Verifying both that the requirements meet the needs of the customer and that the device meets the requirements is equally important. Identifying the requirements of the software system is difficult since it requires understanding the process or activity to be computerized as well as the needs and desires of the eventual users or operators of the system.

The document must contain all of the requirements that the device will be expected to meet. It needs to be modifiable and useable at all times. Documentation needs to change as the requirements are changed and adjusted. Change can occur during the specification phase when requirements are changed and/or modified; during the design phase when implementation requires a different approach that alters the requirements specification; through the change and maintenance phases when additional requirements are introduced to the device or changes are made to how the device operates. During this iterative process, handle the software requirements specification document like an approved final document. Incorporate changes using a change control system. Ideally, change control starts when the design phase begins. Archiving these changes would help during the maintenance phase, when adjusting the software either to provide additional functions or correct bugs found in the system. Troubleshooting and modification of software programs must be possible from the documentation.

The documentation method should develop an audit trail as changes occur. The evolution of the requirements specification can be as important as the final product. As the life cycle of the product continues with software and requirements updated and modified, change the software requirements specification document to reflect the current requirements of the system. Copies of past specification documents must be kept in a history file. The history file serves three purposes: 1) to show the evolution of the software; 2) to give the manufacturer or user information on when the specifications changed (beginning with which serial or lot number); and 3) to prevent future changes from the same mistakes made in the past. Historical records also need to trace problems with the existing devices to show how the devices were manufactured. The developer also can use the old documentation if a change was incorrect and he needs to fall back to a previous version.

Change control requires that a baseline of all approved documents be maintained and changes to these documents be controlled. Prior to coding, the change(s) in a requirements specification are translated to the programmer within the documentation. However, if during coding the programmer changes the requirements, he must incorporate this change into the requirements specification document. Any changes to the documentation should be reviewed at appropriate levels to assure that the change(s) do not affect other areas or the intent of the original specification.

The software requirements specification document should be divided into sections. ANSI/IEEE Std 830 - 1984 [15], Fairley [4], and DOD-STD-2167A [17] are three references that describe ways the information can be arranged. All documents need to be readable and understandable to the designers and the engineers, as well as, the users, customers, and marketing people. Use whatever format accomplishes these goals. Procedures for functional validation and testing of the software are developed from the software requirements specification [1, 6].

### A Model for Software Requirements Specification Document

A model for use for medical device software requirements specification document is discussed below [7]. This model combines the ANSI/IEEE and Fairley models and is designed to meet the regulatory requirements of the Food and Drug Administration. The model is not endorsed by the Food and Drug Administration nor proposed as the only model that could be used to meet the regulatory requirements. The model can also be used to analyze existing software requirements specifications documents.

#### Introduction and Scope

This section can be divided into three subsections. Each subsection will be described using questions to be answered by those sections.

Purpose or Identification The purpose of the document should be stated with regard to the definition of the medical device software (device) to be described by the software requirements specification document. This statement should be brief but complete.

1. Define the device.
2. Where will the device be used or operated?
3. Is the device critical or non-critical?
4. Is the device self-contained or an adjunct to another device?

Product Overview A brief summary of the device and the environment in which the device operates should include information on who will operate the device. This section is meant to aid the reader in understanding the specific requirements section better. Block diagrams can be used to show the interconnections and external interfaces for the device without imposing design solutions.

1. Who will use the device?
2. Summarize product features and requirements.
3. What are the principal interfaces for the device?
4. What hardware and peripheral equipment will be used with the device? (overview only)

Processing Environment The development, operation and maintenance of the device should be described as it relates to the processing environment. The processing environment is the development, operating, and maintenance environments in which the device will operate. This section will contain information that can later be expanded in the User Manual.

1. Who will operate the system?
2. Who will/can modify and/or update the program?
3. Brief description of development process,

including testing of the device prior to and after marketing.
4. What post marketing support will be available?

External Interface Requirements

Four types of interfaces should be discussed in terms of the software program for the medical device. Each section may begin with a summary of the requirements for that interface and then go into greater detail. Explanations of the specific requirements or design constraints are helpful in guiding the later design of the system. Data diagrams showing data flow into and out of the device can help to graphically depict the system's interaction with the outside world. The data dictionary can be used to explain the data flows. Information in the data dictionary should include the name of the data item or control flow, what the purpose of the element is, and any sub items of which the element is composed. The data dictionary will probably be an attachment to or exhibit of the document and will be later expanded in the design document.

User Interface A list of what the user will see and use should comprise this section. The following questions should be answered:

1. Who are the users of the device? Are they exclusive operators or occasional users? What training or experience will enable them to use the device?
2. Will the system be self-explanatory?
3. What are the user commands?
4. What input will the user give to the program?
5. What output or reports will the device produce?
6. What is the timing between inputs and outputs?

Hardware Interfaces Discuss how the software will interact with the hardware part of the medical device. Describe what the hardware is, in general terms. If the system is embedded in a specific device, this should be explained. If both the hardware and software systems are being designed concurrently, then information on how the two will work together should be explained.

Software Interfaces Discuss uses of other required software products, either commercial or firm designed, with the new software. Include information on the links between the software, such as passing information, and operations that will be done outside of this new software program. Questions that need to be answered are:

1. What is the other software package by name, version, source, and location?
2. Why is the interface necessary?
3. What type of documentation does the other software have?
4. What information will pass between the programs?

Communications Interface If the software will communicate with a network, then its interfaces to that network must be specified.

1. Why does the software connect to a network?
2. What information will be exchanged?

Medical Device Functions

Explain the detailed functions of the medical device. This section should contain all of the information that the software designer will need to develop the software. Express the functions in formal notation. The format of this section should allow the reader and the designer to understand the software's functions, inputs, outputs, etc. All information discussed in the eight sections below is necessary.

Summary of Functional Requirements Present product requirements in a manner so anyone reading the document will understand the functional requirements of the medical device. Reasons for specific requirements should be explained in this section. This section should make the functions understandable to the first time reader. The language should be less formalized. This section summarizes the information in the specific requirements section and shows the interrelationship of all the requirements for the medical device software.

Specific Functional Requirements Formalized language, equations, and flow diagrams should be used to identify the requirements. Each functional requirement should have its own subsection with four subparagraphs:

Introduction: The introduction should describe the purpose of the function. All background material necessary to clarify the intent of the function should be included.

1. What does the function do?
2. Why is this function necessary?

Inputs: Data and control inputs into the function, including units of measure, timing, quantities, and valid ranges should be described. Any external inputs such as operator commands should be described.

Data Processing: Processing of the input data, operations performed, intermediate or local parameters used, and outputs obtained should be fully explained. Validity checks on the inputs, sequencing and timing of the operations, responses to abnormal situations such as overflow, error handling etc., parameters affected by the operations performed, validity checks on the output should each be explained. Algorithms, equations, logical operations, etc., that are used to change the inputs into outputs should be included here.

Outputs: Descriptions of the outputs from the function should be described in the same terms as the inputs. Error messages, disposition of illegal or out of range values and the destination of each output should be detailed.

Performance Requirements Performance requirements of the software system should be described here. They can be further divided into static and dynamic. The requirements include the internal numerical constraints and human interaction with the software. They must be presented in quantifiable terms and methods to verify the requirements should also be included. Static performance requirements include the capacity of the files, number of simultaneous system users, size of internal tables, constraints of primary and secondary memory, etc. Dynamic performance requirements include the number of tasks that will be performed within a given time period, or the amount of data expected to be processed within a given time period. This is stated for the entire software package. Any requirements that are specific to a function, should be stated in the processing subparagraph for that function.

1. How will the device perform?
2. How long will it take to do its operation.
3. When will it need information given to it by another user or program?
4. How many operators can use the device at a time?

Exception Handling The device may not always be operated in an acceptable environment. When unacceptable events occur, error messages and their resolution must be delineated. A table of exception conditions and responses is one way to represent this to the reader. As many of these as can be foreseen should be handled. As the program is debugged, this part of the documentation may change and expand to incorporate other conditions that come up during debugging and initial test runs.

Design Constraints The design of any medical device may be subject to standards governmental, industrial, or voluntary. The medical device must comply with the GMP's and may have performance requirements that are required by the FDA (ie. x-ray performance standards) or recommended by other standards organizations (UL, ANSI). These standards may require certain data elements, report forms, user interfaces, accuracy, and precision limits, etc. Within the context of these constraints the software must be designed. If the software is to work with already existing hardware, then the characteristics of the hardware configuration must be included in the requirements document.

Attributes All software has characteristics that add requirements to the system. Some of the attributes that may need to be addressed are the availability of

the system; security from unauthorized access, use, modification, or destruction of the software; maintainability; transportability or conversion of the software between environments; etc.

Design Enhancements Medical devices must be designed to accommodate an ever changing medical environment. If during the initial phase of determining the requirements of the device, additional requirements, design constraints, or foreseeable modifications are already known, each should be included here. The designer of the software may modify his design so that these changes can be incorporated more easily at a later date.

Acceptance Criteria The acceptance tests that will be performed to assure that the functions are met should be discussed. Acceptance testing should be based on the functional requirements of the system. Documentation standards for the tests performed may be described or their location may be referred to. Acceptance criteria should quantitatively verify the functional requirements stated in the specific requirements section. Any audits of this documentation and what will be covered during them should be delineated.

Definitions

Any terms that may be new or defined differently should be defined here. Definitions should be complete and precise. Common terms may also be defined for clarity and for the reference of non-technical reviewers of the document.

## CONCLUSION

The software requirements specification document is a necessary and cost effective part of designing medical devices. Any company, large or small, who buys, develops, or manufactures software for or as part of a medical device can use this model. The software requirements specification document must be updated throughout the life of the software. During the maintenance phase, the document helps the company understand the process by which the device was designed so that change or modification is accomplished smoothly. Thoughts of the original specifiers are lost if the only documentation kept is the code and test documents.

The software requirements specification document describes the behavior of the medical device software in terms that the manufacturer/owner can understand. It will help the designers and programmers understand the requirements up front and should avoid duplication, arguments, and inconsistency with requirements during their parts of the software life cycle. This document should be a good basis for test plan development. The software requirements specification document, if properly completed and updated, will be useful to the maintainers of the software long after all the original personnel, who developed it, are no longer available for consultation.

## BIBLIOGRAPHY


[1] R. D. Berkland, "Validatable Software Design", Pharmaceutical Technology, vol 15, no. 9, pp. 160-165, September, 1991.

[2] I. P. Cooper and B. F. Mackler, "Regulation of Diagnostic Software: Preparative Steps", Medical Device and Diagnostic Industry, vol. 7, no. 10, pp. 38-42, October, 1985.

[3] A. M. Davis, "A Comparison of Techniques for the Specification of External System Behavior", Communications of the ACM, vol. 31, no. 9, pp. 1098-1115, September, 1988.

[4] R. E. Fairley, Software Engineering Concepts. New York: McGraw-Hill, 1985.

[5] D. I. Good, "Cost-Effectiveness", ACM Software Engineering Notes, vol. 11, no. 11, pp. 82, November, 1986.

[6] P. A. V. Hall, " Relationship between Specifications and Testing", Information and Software Technology, vol. 33, no. 1, pp. 47-52, January/February, 1991.

[7] J. V. Halvorsen, "A Proposed Software Requirements Specification Document for the Medical Device Industry", Masters Thesis, University of Central Florida, College of Engineering, May, 1992.

[8] M. F. Houston, "Designing Safer, More Reliable Software Systems", presented to the Medical Design and Manufacturing Conference, New York, NY 1988.

[9] J. S. Kahan, "FDA Regulation of Computer Software as Medical Devices", Medical Device and Diagnostic Industry, vol. 7, no. 3, pp. 51-54, March 1985.

[10] R. M. Kosmala, "The Relationship between Functional Requirements and Software Design", Pharmaceutical Technology, vol 14, no. 11, pp. 66-68, November, 1990.

[11] P. G. Neumann, "Flaws in Specifications and What to do about Them", Proceedings of Fifth International Workshop on Software Specification and Design, Pittsburgh, PA, 19-20 May 1989, pp. xi-xv.

[12] D. L. Parnas and P. C. Clements, "A Rational Design Process: How and Why to Fake It", IEEE Transactions of Software Engineering, vol. SE-12, No. 2, February, 1986.

[13] D. L. Ripps, "Developing Real-time Requirements", EDN, vol. 35, pp. 223-228, October 11, 1990.

[14] J. V. Sroka and R. M. Rusting, Medical Device Computer Software: Challenges and Safeguards", Medical Device and Diagnostic Industry, vol. 14, no. 1, pp. 63-65, 165-167, January, 1992.

[15] ANSI/IEEE Std 830-1984, IEEE Guide to Software Requirements Specifications, Institute of Electrical and Electronic Engineers, New York, 1984.

[16] ANSI/IEEE Std 828, IEEE Standard for Software Configuration Management Plans, Institute of Electrical and Electronic Engineers, New York, 1983.

[17] DOD-STD-2167A Military Standard Defense System Software Development, Department of Defense, Washington, D.C., February, 1988.

[18] Food, Drug, and Cosmetic Act, 21 USC 321[h], 1990.

[19] Medical Device Good Manufacturing Practices Manual, Fifth Edition, U.S. Government Printing Office, Washington, D.C., 1991.

[20] 21 Code of Federal Regulations, parts 800-1299, U.S. Government Printing Office, Washington, D.C., 1991.