Applying Requirements Management to Medical Devices Utilizing Software

by Dean A. Leffingwell, Don R. Widrig and Wayne T. Morrissey

The Regulation of Medical Device Software

The state of the art in medical device software development has undergone enormous changes in the past decade. In the last 10-15 years of medical software regulation, the FDA has become aware that there was significant room for improvement. Indeed, the FDA found that approximately 44 percent of the quality problems that led to voluntary recalls during this period were attributed to errors or deficiencies that were designed into particular medical devices rather than having been inserted in the manufacturing phase. Furthermore, it seemed clear that many of these errors could have been prevented by adequate design controls.[1]

In an effort to normalize the US standards with the evolving world market places, and in an effort to improve regulation of medical device development, enabling legislation was passed in 1990 to give the FDA the necessary scope to regulate the development of medical device software. The enabling legislation, contained in the Safe Medical Devices Act (SMDA) of 1990, was the impetus for the FDA to drastically revise its oversight into design development of medical devices containing software.

The major result of passage of the SMDA has been a drastic revision of the old GMP regulations. The new GMP regulations have only recently been approved and the impacton medical device software is enormous.

In brief, the new GMP regulations, now referred to as the Quality System Regulation (QSR), take effect with a transition period from June, 1997, through June, 1998[2]. Prior to the end of this transition period, all development of Class II and Class III medical devices automated with software must be moved to full compliance with the development process standards outlined in the QSR.

The good news is that the QSR has excellent guidance on the establishment of modern software development practices. Indeed, these mandated practices have been carefully worked out by the regulations in the QSR so as to conform to standards which you may be familiar with. QSR was specifically crafted to conform to the IEEE Software Engineering Standards[3] or the worldwide standards such as may be found in ISO 9001, IEC 601 or EN 46001.

Further good news may be found in the QSR in that no specific standard is absolutely required. Thus, you can model your own development processes after existingstandards and you can adjust the model to suit your own particular development needs.

Since 1984, the FDA has been struggling with the notion of establishing design controls, including safety and effectiveness, as part of the overall quality process. Now, with the advent of the QSR, a major step has been taken to insure quality of software development in a medical device. In the next part of this paper, we shall begin exploring how a Requirements Management tool such as RequisitePro™ can help you efficiently automate many of the tasks prescribed by the QSR as part of a development process for world-quality medical devices and products.

Part I of the following sections outlines the structure and documentation required for a medical device software development process. In Part II, we will explore a sample medical device development plan and examine the features offered by the RequisitePro tool and support that the tool provides in automating and managing the requirements for the device.

---

[1] FDA, *Medical Devices; Current Good Manufacturing Practice (CGMP) Final Rule; Quality System Regulation*. p 52602.

[2] Ibid. p 52604.

[3] IEEE. *IEEE Standards Collection, Software Engineering*, IEEE, New York, NY. 1994. The collected set of standards will be referred to throughout this paper.

Part I. Formulating An Acceptable Development Plan

Beginning in mid-1997, the Quality System Regulation (QSR) mandates the establishment of a wide-ranging and well-structured plan for the development of software for medical devices that depend on software for their operation. The key word here is the plan. While the QSR mandates the establishment of a plan, it does not describe the contents of such a plan or the process by which the software is developed. To obtain this insight, it is necessary to become familiar with another new FDA guidance document published by the Office for Device Evaluation (ODE) group, "ODE Guidance for the Content of Premarket Submission for Medical Devices Containing Software (draft document)." The document is a guidance document and does not have the force of regulation. However, the document covers a large number of practical suggestions for the development of a plan. We will refer to this document as the "ODE Guidance" document (OG).

Appendix A of the OG defines a number of distinct stages in the software development lifecycle. Appendix A defines the following major areas of the lifecycle:

- Requirements Analysis and Specification
- Architectural Analysis and Specification
- Design and Development
- Verification
- Validation
- Configuration Management and Change Control
- Independent Verification and Validation

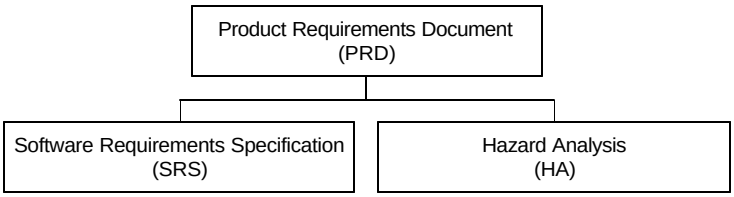
In this whitepaper, we will explore the highlights of each area in order to establish an overview of the processes and documents required for each stage. In Part II of this white paper, we will use the RequisitePro toolkit to automate and support the requirements management activities which support these stages.

Requirements Analysis and Specification

Section A.2 of the OG recognizes the importance of the need to identify and analyze end-user functional and performance requirements for the product. The Requirements Management team at Rational Software is an industry leader in this area and has published papers and courses in support of the vital importance of collecting and analyzing requirements information.[4]

In order to organize and manage this effort, many medical device projects will probably be well served by defining and using three different types of documents to collect and define the requirements: the Product Requirement Document (PRD), the Software Requirement Specification (SRS), and the Hazard Analysis (HA). These documents form the top of an implementation documentation structure that initially appears as shown in Figure 1.

Figure 1, Initial Implementation Document Setup



Product Requirements Document (PRD)

The PRD collects, analyzes, and defines the features of the product and the user needs that the device addresses. While no widely adopted standard for such a document exists, RequisitePro offers a template for a Product Requirements Document as a starting point for managing your project's requirements.

[4] Leffingwell, D., and A. Davis, *Using Requirements Management to Speed Delivery of Higher Quality Applications*, Rational Software TR0001, 06/96.

The PRD is commonly initiated through your organization's marketing department working in conjunction with clinical specialists. It offers the marketing department a palette to record high-level user needs and establishes the clinical claims for the device.

It should also serve as the organizing element to focus safety features, conformance to standards, clinical claims, and even subsequent marketing collaterals.

Software Requirements Specification (SRS)

Typically, the requirement gathering process starts out very abstractly and culminates in a series of high-level product features. These features are recorded and managed in the PRD discussed above. The SRS is written to respond to the software-fulfilled behaviors that are specified in the PRD. In modern medical devices, software may occur in an embedded microcontroller that operates the device, software may occur as part of an interface to other devices, or the device may also contain external, stand-alone, software for post-processing of data. Regardless of the function of the software, all requirements for the software should be specified in one or more SRS documents.

The software requirements provide a detailed specification of exactly what the software must do, and not how the software is to be designed or implemented. A list of principles to follow when you are documenting the software requirements may be found in Chapter 3 of *201 Principles of Software Development*[5]

The key points to recognize when writing the SRS include[6]:

- The requirements should be complete, consistent, and as unambiguous as practical.
- Every requirement should be traceable back to one or more features in the PRD and traceable forward to lower level requirements, test cases, and implementation modules.
- Every requirement should be assigned a "tag" so that it can be identified, tracked, and managed as a separate element of the project.

The IEEE offers an excellent set of discussions and templates for an appropriate SRS. Refer to IEEE 830-1993, *Recommended Practice for Software Requirement Specifications* for further information. The RequisitePro tool incorporates the IEEE recommendation into its basic templates of document styles, thus allowing you to quickly lay out the document and begin entry of requirements.

Hazard Analysis (HA)

An important early document in the design process is the Hazard Analysis (HA). The FDA is focusing on the HA as a key element in the improvement of medical device quality. Indeed, the OG devotes section 2.8 entirely to the question of Risk Management and Hazard Analysis. A Hazard Analysis is:

> "the detailed examination of a device from the user and patient perspectives. Its purpose is to detect potential design flaws—possibilities of failure that could cause harm—and to enable manufacturers to correct them before a device is released for use."[7]

As noted, HA requires the designer to consider various classes and types of errors that may occur in the product yet to be constructed. As each potential hazard is documented and examined, the HA document allows the designer to document the potential hazard and then suggest design strategies and specific functional requirements intended to alleviate the hazard.

---

[5] Davis, A., *201 Principles of Software Development*, New York, NY. McGraw-Hill, Inc., 1995.

[6] Leffingwell, D., and A. Davis, *Using Requirements Management to Speed Delivery of Higher Quality Applications*, Rational Software TR0001, 06/96. p 8.

[7] Bill J Wood and Julia W Ermes. *Applying Hazard Analysis to Medical Devices*, Medical Device & Diagnostic Industry magazine, 01/93.

With the advent of the draft OG and the CGMP, new and more sophisticated approaches to Risk Analysis have been added to the traditional Hazard Analysis approaches.

Appendix B of the OG suggests approximately 20 topics which require your scrutiny while conducting the HA. The FDA has found that these topics pose special risks when applied to the software development for medical products and are worth consideration for possible inclusion in the HA.

In subsequent stages of the product development lifecycle, the HA document will serve as a living document to record both the potential hazards and the risk mitigation strategies that have been defined to prevent the hazard from occurring. System validation will later refer to this document to confirm that all anticipated hazards have been completely addressed and resolved.

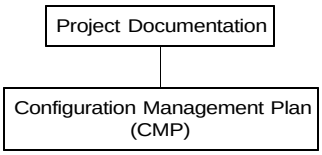
Architectural Analysis and Specification

Appendix A.3 of the OG begins the implementation process. In this phase, the functional and safety requirements are allocated to the hardware and software aspects of the product. Tradeoff studies may be performed to determine the most effective implementation approach.

The key document produced in this part of the development lifecycle is the Configuration Management Plan (CMP). Excellent discussions of the concept, layout, and use of the CMP may be found in existing standards IEEE 828-1990, *Standard for Software Configuration Management Plans*, and IEEE 1042-1987, *Guide to Software Configuration Management*.

The CMP stands outside of the implementation document tree since this document is a project-wide guidance document. The initial project-wide documentation tree appears as shown in Figure 2.

Figure 2, Initial Project Documentation Setup

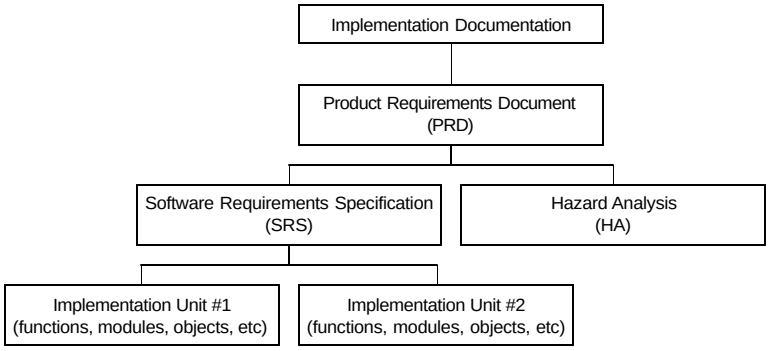


Design and Development

Appendix A.4 of the OG discusses the translation of software requirements from the SRS into source code. In order to standardize the implementation practices, the OG recommends the use of style guides, coding standards, etc. Walkthroughs and bench testing are recommended practices during this activity. Typically, software is organized around some type of implementation unit such as modules of code, subroutines, object classes, etc. Thus, another set of documents is either explicitly or implicitly being built, that of the implementation units.

As the implementation documentation becomes available, you should add it to the implementation documentation structure as shown in Figure 3.

Figure 3, Implementation Documentation



The OG recommends that the developer maintain a strict audit trail between the implementation units and the specifications and HA for that implementation. We will discuss how to do this in Part II when we discuss traceability.

Verification

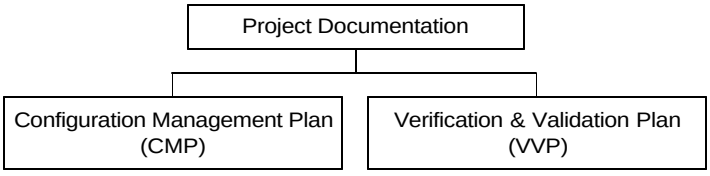
Appendix A.5 of the OG covers the Verification activities. The IEEE defines "verification" as:

> "The process of evaluating a system or component to determine whether the products of a given phase satisfy the conditions imposed at the start of that phase."[8]

That is, the Verification activity is largely a paper-based activity that requires you to confirm that each stage of the development (e.g., a software implementation of one or more requirements) conforms to the requirements defined in the previous stage. In order to have a method to perform this Verification, you need a plan.

A well-organized project will include a *Verification and Validation Plan* (VVP). As usual, the IEEE offers excellent guidance for setting up a VVP in IEEE 1012-1987, *IEEE Standard for Software Verification and Validation* and IEEE 1059-1993, *IEEE Software Guide for Verification and Validation Plans*. Note that the VVP is a project-wide document that establishes the rules for Verification testing (as well as Validation testing). Thus, this document can "stand to the side" as a project document, similar to the Configuration Management Plan (CMP) mentioned earlier. The project documentation tree then appears as shown in Figure 4.

Figure 4, Project Documentation



The OG recommends that the developer maintain a strict audit trail between the Verification activities and the product specifications and HA for that implementation. We shall return to this topic in Part II when we discuss traceability.

Validation

In a similar manner, Appendix A.6 of the OG recommends various Validation activities. The IEEE defines "validation" as:

> "The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements."[9]

In other words, you need to confirm that the implemented component actually works to specification. Normally, the major means to satisfy this activity is testing. Once again, a Validation plan is needed. Traditionally, the Validation plan is included as part of the V&V plan (VVP) discussed earlier.

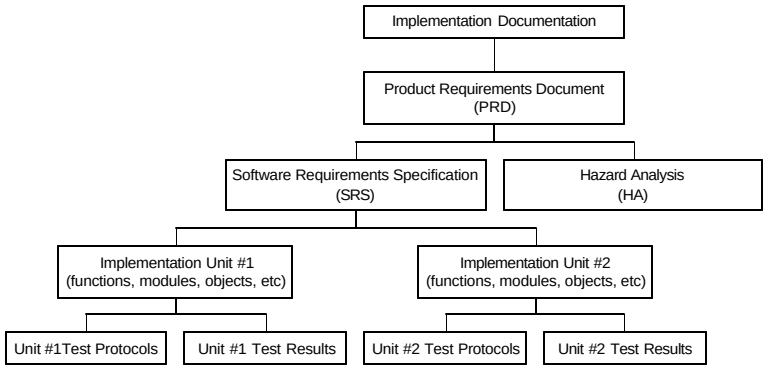
Validation activities go hand in hand with testing. But, what does a good test plan look like? Fortunately, the IEEE has an answer. IEEE 829-1983, *IEEE Standard for Software Test Documentation* provides extensive coverage on the establishment of a test methodology, conducting tests, reporting results, and resolving anomalies.

[8] IEEE. *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Standards Collection, Software Engineering, IEEE, New York, NY. 1994.

[9] Ibid.

Note that software testing should confirm the correctness of the units under test from two perspectives: 1) the unit meets the implementation element's specification and, 2) the unit meets its governing requirements. That is, the tests should not only confirm correct operation of the unit, they should also confirm that the original specifications have been met. The test documents form part of the implementation documentation. Allowing for test documents, the implementation documentation tree should appear as in Figure 5.

Figure 5, Implementation Documentation (with Testing)



The OG recommends that the developer maintain a strict audit trail between the Validation/testing activities and the specifications and HA for that implementation. This audit trail is provided via the mechanism of requirements traceability.

Configuration Management and Change Control

Appendix A.7 of the OG specifies issues relating the change management. Note that you have already allowed for a Configuration Management Plan (CMP) in an earlier step. Now, you need to make sure that the CMP is comprehensive in the area of managing change.

Change is the norm in modern software development projects. To manage change effectively, you must:

- Recognize that changes come from many sources and know what those sources are
- Create an explicit process that will review and analyze all requests for change
- Baseline your system so that you can recognize changes that occur

Fortunately, these issues are well structured in the CMP you have developed earlier. In Part II of this paper, we will consider techniques and mechanisms for analyzing and managing the impact of change.

Independent Verification and Validation

Appendix A.8 of the OG recommends Verification and Validation (V&V) activities that are performed by "outsiders." Typically, those people immersed in the day-to-day minutia of actually implementing a software project are prone to "blind spots" which may conceal a potential problem with the medical product's design or implementation. Thus, the FDA recommends the use of technically qualified people to conduct independent reviews of the project. In principle, these reviews are no different than the V&V activities you have performed yourself. Indeed, it is quite efficient to have the outside reviewers be familiar with the current VVP but you should expect these reviewers to add and revise the VVP as new areas are explored.

As in the case of earlier V&V activities, you should plan on maintaining an audit trail between the V&V activities (and related documents) and the top-level specifications and HA. We shall return to this topic in Part II when we discuss traceability.

Level Of Concern

The issue of the Level Of Concern is independent of the lifecycle segments mentioned above. Level Of Concern (LOC) is of particular interest in the OG. Simply put, LOC is concerned with the identification of the consequences of device failures and their relationship to patient safety. That is, if the device fails, will it seriously harm the patient, offer minor harm to the patient, or present minimal harm to the patient? LOC is a complex issue that is not

completely resolved in the draft OG and, in fact, resulted in a series of suggested draft approaches which the FDA may take for final publication of the OG. [10]

Traditional approaches to LOC tend to aggregate all of the device aspects into a single LOC assessment. In the traditional approach, the most safety-critical features of the device are considered and used to establish an LOC assessment. The disadvantage of the aggregation approach is that a single high-safety aspect of the device forces a high LOC which, in turn, forces the entire device and all its component parts to be treated as a high LOC issue. In fact, such devices typically exhibit many features that are not a high LOC and which could be developed on a much more relaxed LOC basis.

In order to avoid wasting resources in the development of low-LOC portions of the product, we will become more selective in the handling of requirements that may have different LOCs. In this manner, we can manage portions of the project differently in a manner consistent with the assigned LOC for each segment.

To effectively segment the project, we will assess and assign LOCs on a feature-by-feature basis at the PRD level and a requirement-by-requirement basis at the SRS (and HRS) level. As we shall see in Part II, RequisitePro provides a feature, requirement attributes, for the handling and management of different LOCs in the project.

[10] FDA/ODE, *ODE Guidance for the Content of Premarket Submission for Medical Devices Containing Software* (draft 1.3, 12 Aug 1996). Attachment 2.

Part II. Implementing Software Requirements Management As Part Of Your Development Plan

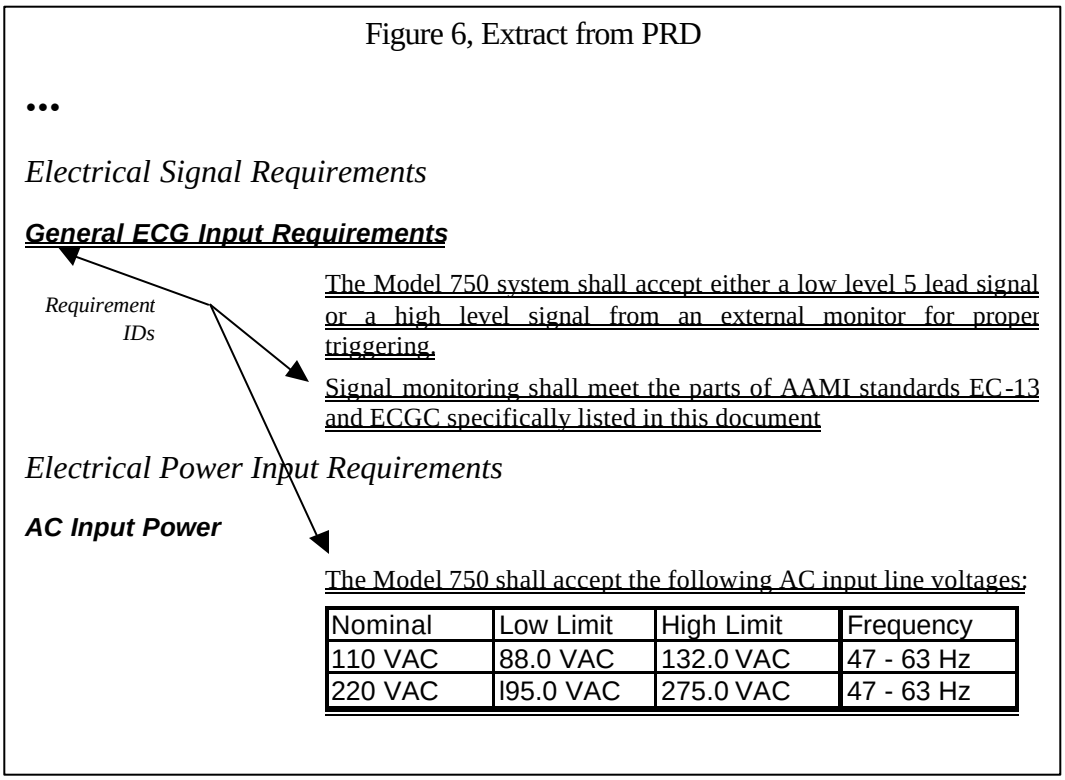
The Project

To demonstrate the possibilities offered by the RequisitePro Requirements Management tool, we will examine software development processes in an abbreviated medical product development project. The device under consideration is an imaginary device referred to as a Reverse Angioplasty Pump (RAP).

Create A Product Requirements Document (PRD)

To properly define the clinical and marketing features for the product, the RequisitePro tool is used to create a Product Requirement Document (PRD). RequisitePro provides a template for this document which can be modified to suit your specific device needs. RequisitePro interacts seamlessly with Word to allow you to create the initial PRD. In addition, the tool allows the document editors to identify specific Product Requirements (PRs) within the document. A small extract from the PRD appears as shown in Figure 6.

Figure 6, Extract from PRD



Notice that the RequisitePro tool allows you to automatically highlight requirements via a double-underline or via a user defined style. Highlighting is a powerful visual aid that is recommended to help the document reader quickly focus on specific behavioral issues, performance issues, safety issues, etc.

Figure 6 identifies both hardware and software features but, to simplify this white paper, we shall treat all of the features as leading to software requirements. There is no loss of generality in doing so, but a real project would probably make a more refined distinction between hardware and software features (and the documents that record the features).

RequisitePro automatically assigned a unique identifier (PR4, PR5, etc) to each product requirement as it was identified; thus conforming to the FDA guidance referred to in Part I.

RequisitePro should also be used to assign and maintain a set of attributes and values for each PR in the PRD.

For now, we will use the Attribute feature to classify the Level Of Concern (LOC) on a feature-by-feature basis. As each feature is defined, we will insert an attribute value, the Concern attribute that we have defined for PRs. We can use RequisitePro to define this attribute and a list of acceptable values. Then, as the features are defined, you can assess the LOC and record your assessment in the Concern attribute for the feature. RequisitePro offers easy-to-use selection and sorting capabilities to later select only features that have a selected LOC, or some other combination of attributes and attribute values that is of interest to the team.

At any point in the project, RequisitePro can be used to print the document or display a data view that provides a current listing of all PRs defined. Optionally, this view can be filtered or sorted based upon various attributes that are of interest from a particular perspective. An example of such a view is shown in Table 1.

Table 1, Extract of Sample PRs from PRD

Requirements	Status	Concern	Difficulty	Class
•••	•••	•••	•••	•••
PR4: General ECG Input Requirements The Model 750 system shall accept either a low level 5 lead signal or a high level signal from an external monitor for proper triggering.	Approved	Moderate	High	Diagnostic
PR5: Signal monitoring shall meet the parts of AAMI standards EC-13 and ECGC specifically listed in this document	Approved	Critical	High	
PR6: The Model 750 shall accept the following AC input line voltages:	Approved	Critical	Medium	Ease of Use
•••	•••		•••	•••

Create A Software Requirements Specification (SRS)

In a similar manner, you can use RequisitePro to create, edit, and maintain the Software Requirements Specification (SRS) for the product based upon templates provided for this purpose. A small extract of the document appears as shown in Figure 7.

Figure 7, Extract from SRS

External Communications Device Interface

The modem port shall be initialized on system power up or system reset.

Upon initialization, the modem port is prepared for transmission of diagnostic data at 14.4k baud.

Protocol shall be no parity, 8 data bits, 1 stop bit.

The modem shall also be initialized to "auto answer".

Upon command from the front end, the system assembles and initiates transmission of a frame of diagnostic data to the modem port.

•••

As in the case of the PRD, we have chosen to highlight the Software Requirements (SRs). A s before, RequisitePro automatically assigned a unique identifier (SR1, SR2, etc) to each software requirement as it was identified, thus conforming to the FDA guidance referred to in Part I.

RequisitePro should also be used to assign and maintain a set of attributes and values for each requirement in the SRS. Note that the attributes and their values can be independently assigned for each type of requirement. Note that we have implemented the Level Of Concern issue via the same attribute concept as used in the PRD but we have chosen to define other attributes differently than the PRD attributes. A sample view of SRs and some of their attributes is shown in Table 2.

Table 2, Extract of Sample SRs from SRS

Requirements	Status	Concern	Priority	Assigned to
SR1: The modem port shall be initialized on system power up or system reset.	Approved	Minor	Low	Team B
SR2: Upon initialization, the modem port is prepared for transmission of diagnostic data at 14.4k baud.	Approved	Minor	Medium	Team C
SR3: Protocol shall be no parity, 8 data bits, 1 stop bit.	Proposed	Minor	Medium	Team B
SR4: The modem shall also be initialized to "auto answer".	Approved	Minor	High	Team B
SR5: Upon command from the front end, the system assembles and initiates transmission of a frame of diagnostic data to the modem port.	Proposed	Moderate	High	Team B
•••	**•••**	**•••**	**•••**	**•••**

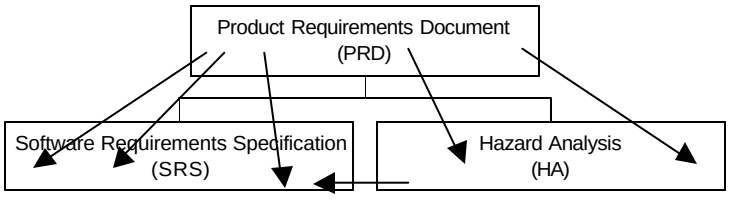
As in the case of the PRD, you can use RequisitePro's query engine to sort, extract, and manage SRs that have a specified set of attribute values.

Create A Hazard Analysis

In a similar manner, you can use RequisitePro to create, edit, and maintain the Hazard Analysis (HA) for the product. Using RequisitePro, a Word document detailing the software (and possibly hardware) safety requirements should be created.

Once the PRD, HA, and SRS have been generally completed (it is not necessary to wait until the "final" versions are approved), you should begin the process of relating the documents. The objective is to understand how the elements of one document relate to the elements of another document as shown in Figure 8.

Figure 8, Initial Traceability



The individual elements of each document should be linked to appropriate elements in the other document. That is, you should now relate each SR entry in the SRS to its governing PR entry in the PRD. Or, conversely, you should match each PR entry to all of its governed SR entries. Notice that this matching may be one-to-many, many-to-one, or many-to-many.

In a similar manner, the HA entries should be linked into their respective relationships. RequisitePro does not require a strict hierarchical structure for the relationships. Therefore, both "vertical" relationships such as PR-to-SR and "horizontal" relationships such as HA-to-SR are permitted. Non-hierarchical relationships are a normal part of most development projects and no special characteristics should be implied by non-hierarchical relationships.

Regardless of the relationship, it is important to link associated items together. This linking process is referred to as traceability.

Traceability

A significant factor in quality software implementation is the ability to trace the implementation through the stages of specification, architecture, design, implementation, and V&V. Indeed, the ability to track relationships and relate these relationships to the issue of change management forms a key thread throughout the new OG[11]. In addition, the Design Controls section of the new CGMP[12], Subpart C of CGMP, makes repeated references to the need to be able to trace the relationship between various work products within the lifecycle of the product's development. IEEE provides two working definitions of traceability:[13]

1) "The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; for example, the degree to which the requirements and design of a given software component match."

2) "The degree to which each element in a software development product establishes its reason for existing; for example, the degree to which each element in a bubble chart references the requirement it satisfies."

A key element of traceability is the definition of what is meant by a "traceability relationship." In RequisitePro, it is convenient to define the relationship in terms of a simple "traced to" and "traced-from" model. For example, we can easily imagine that one or more Software Requirements (SRs) are created in the system in order to support a given feature specified as a Product Requirement (PR). Thus, we can say that an SR is traced-from one or more PRs. Additional meaning can be placed on the relationship from the context of the requirement types that are created. For example, a SR that is traced to a Test Case requirement type, would infer that the software requirement is "tested-by" the test case that it is "traced-to." A class description that is traced-from a SR requirement would imply that the requirement is "implemented-by" the referenced class. In RequisitePro, there is no limit to the number and types of requirement types that can be defined. In addition, requirements of a given type can appear within any document. For example, it is not necessary that only requirements of type SR reside within a document that describes software requirements.

RequisitePro offers a simple user-guided procedure to "point and click" through the relationships that may exist between two elements of the lifecycle. After you have defined the relationships between the PRs and the SRs, RequisitePro can display a matrix version of the relationships between the PRs and the SRs as shown in the example of Figure 9.

Interpretation of the traceability matrix in Figure 9 is straightforward. For example, consider the intersection of PR8 (Remote Data Communications…) and SR1 (The modem port…). At the intersecting cell, the arrow "⇗" indicates that there is a relationship that traces from PR8 to SR1, meaning that SR1 is derived from, or in some way satisfies the feature defined as PR8.

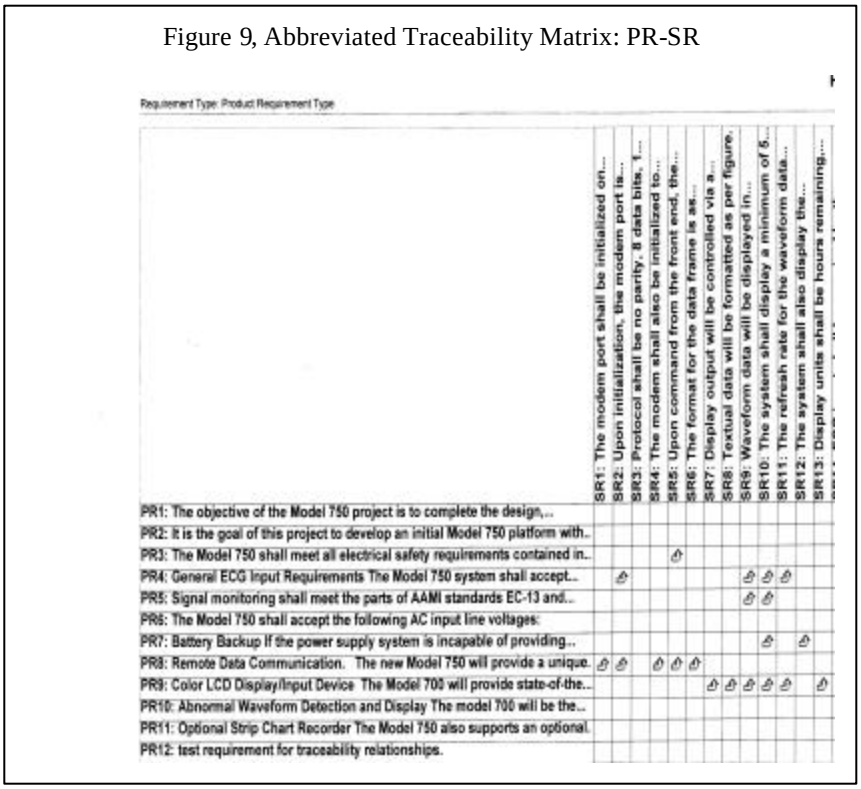
---

[11] FDA/ODE, *ODE Guidance for the Content of Premarket Submission for Medical Devices Containing Software* (draft 1.3, 12 Aug 1996).

[12] FDA, *Medical Devices; Current Good Manufacturing Practice (CGMP) Final Rule; Quality System Regulation*. Subpart C, pp 52657-52658.

[13] IEEE. *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Standards Collection, Software Engineering, IEEE, New York, NY. 1994.

Figure 9, Abbreviated Traceability Matrix: PR-SR



After using RequisitePro to establish all known relationships, an instructive Requirements Management activity, strongly supported by the FDA guidance, is to examine the traceability matrix for two potential indications of error:

1. If the inspection of a row fails to detect any traceability relationships (no "arrows"), then a possibility exists that there is no software requirement (SR) yet defined to respond to a feature required in the PRD. This may be acceptable if, for example, the feature is to be implemented in other than software (e.g., "The case shall be of non-breakable plastic."). Nevertheless, empty rows are potential red flags and should be checked carefully. RequisitePro has a facility to automate this type of inspection.

2. If the inspection of a column fails to detect any traceability relationships, then a possibility exists that a software requirement has been included for which there is no known product feature that requires it. This may indicate a misunderstanding on the role of the SR or it may indicate a weakness in the original PRD, or it may indicate dead code or code that is not in support of the system requirement. In any case, careful checking is required.

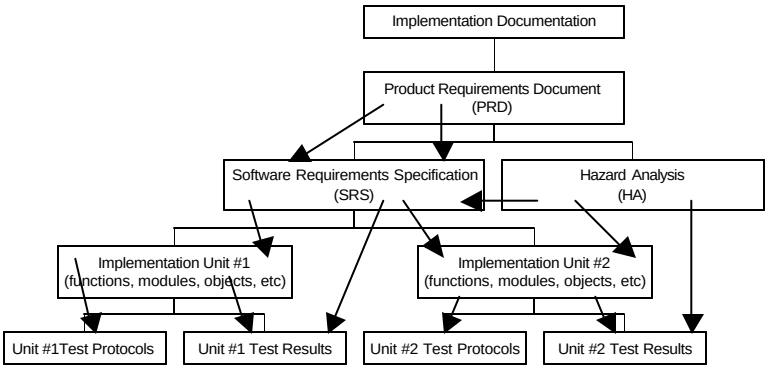
In addition to providing a set of tools to query the relationships you have established, RequisitePro also provides a simple means to store the queries and recall them later. This feature allows you to re-visit the relationships at a later time, perhaps after changes have been made, and quickly re-query the relationships to detect potential trouble spots.

Simple and obvious application of the above techniques will enable you to relate many elements of your project. You should strongly consider linking and relating:

- PRs to SRs
- SRs to Implementation Units
- Implementation Units to Test Plans/Specs/Results
- SRs to Test Plan/Specs/Results
- Hazard Analysis elements (HAs) to PRs, SRs, Implementation Units, and Tests

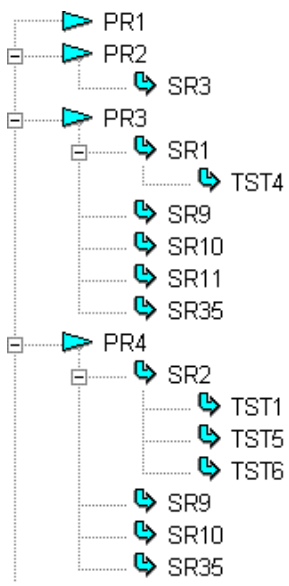
After linking the various elements of the various documents together as suggested above, you should have a relationship setup similar to Figure 10.

Figure 10, Document/Element Relationships



RequisitePro also provides the ability to display the full set of traceability relationships within a project. Figure 11 provides an example of such a "tree" view. Notice that the (partial) tree view allows you to simultaneously view all of the relationships in your project. You should use the tree view to help you comprehend the overall relationships within your project.

Figure 11, Abbreviated View of Traceability Tree



For example, the tree view of Figure 11 reveals that PR3 (a Product Requirement or feature) links to SR1 (a Software Requirement) which, in turn, links to TST4 (a Test Specification).

Once you have linked the elements together, RequisitePro will maintain the linkages for you. You may then use the full power of RequisitePro to examine relationships between the project elements as you desire. A key Requirements Management activity that you will perform regularly is to use the traceability relationships to examine the impact of changes proposed and implemented in your project. This type of activity is referred to in the OG and in the QSR as Change Management.

Change Management

Change Management practices help you to understand and manage three important project development aspects:

1. If an element is proposed for a change (e.g., a single Product Requirement), what are the work consequences of that change? In other words, Change Management helps you address the question of how to determine the amount of rework that may be required if an element is to be changed. The amount of work to effect a change may have significant impact on your project resource planning and workload planning.

2. If an element is proposed for a change, what are the other elements of the system that may be impacted by the change? This topic is of key concern both to your project planning and to the FDA. Experience has taught us that it is inevitably the case that a change to the software will "ripple" into other areas with potentially negative consequences. This is such an important matter in the design and

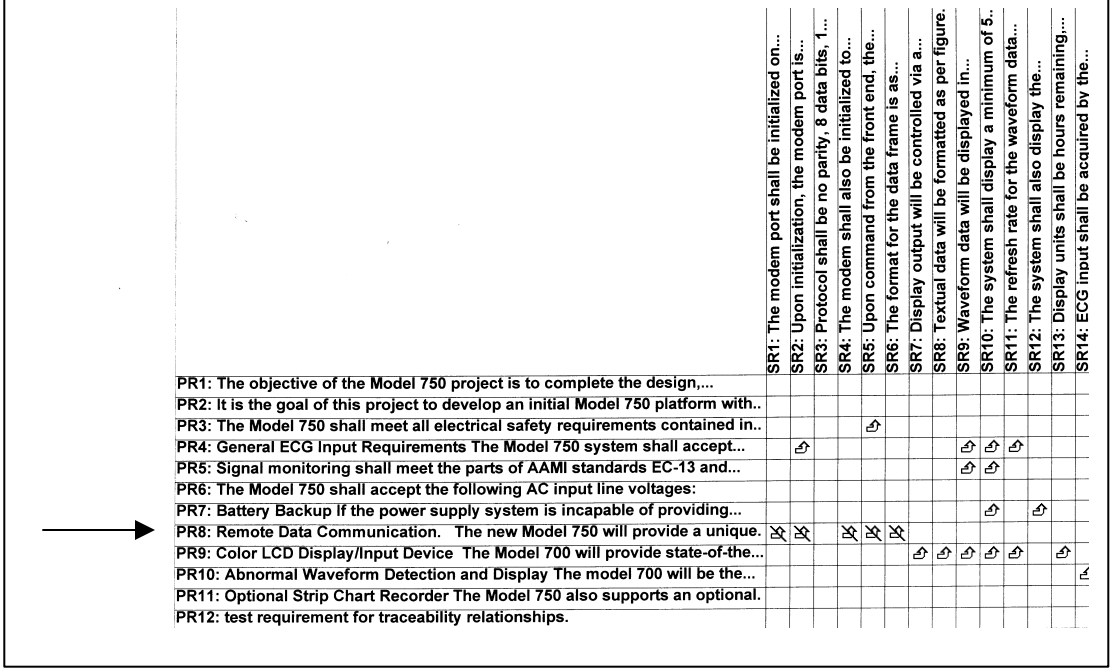
implementation of reliable medical products that the FDA specifically calls for an organized change management procedure as part of the design process.[14][15]

3. Active projects inevitably take wrong turns. It is certain that your project will arrive at a point at which you would like to be able to "roll back" a requirement and examine a previous revision of the requirement. In addition, it would be helpful to remember how and why the requirement was changed. In other words, an audit trail of each requirement is extremely valuable. Not only is this helpful to the project, auditability is also mandated by the FDA as part of the design process.

Elements Impacted By Change

Once you have established the traceability relationships for your project, RequisitePro allows you to use the traceability linkages as a Change Management tool. Let us examine the feature by inspecting the Traceability Matrix previously shown in Figure 9. What if it became necessary to change the wording of PR8 (Remote Data Communication…) to reflect a revised statement of the product feature desired. After using Word/RequisitePro to edit PR8 in the PRD, we find that the Traceability Matrix previously shown in Figure 9 has been automatically altered by RequisitePro and now appears as shown in Figure 12.

Figure 12, Abbreviated Traceability Matrix After PR8 Altered



In Figure 12, notice the diagonal bars that now intersect the traceability arrows in the row corresponding to PR8. These bars are referred to as "suspect links" and are inserted automatically by RequisitePro to warn you that changing PR8 may have an impact on SR1, SR2, SR4, SR5, and SR6.

As the project evolves, you will find that changes are proposed for various aspects of the project. These changes can occur anywhere, from the top-level PRD through specification, implementation, and testing. Whenever a change occurs, RequisitePro will automatically insert the Suspect Link markers to warn you of possible relationships affected by the change. As you inspect the potential interactions, you may find that the affected elements either are affected by the change or they are not. Your Change Management activities usually will involve one of two steps:

[14] FDA, *… Quality System Regulation*. Subpart C, p 52657.

[15] FDA/ODE, *ODE Guidance…*, Appendix A.7.

1. If the affected link is not impacted by the change (e.g., the change to PR8 does not impact SR1), you need only use RequisitePro to clear the Suspect Link. Note that subsequent later changes to PR8 may again set the Suspect Link at some future time.

2. If the affected link is impacted by the change, you may need to rework the affected element. For example, the proposed change to PR8 may require a re-specification of SR2. After editing SR2, you will discover that RequisitePro has automatically added additional Suspect Links to warn you of the potential interactions linked to changing SR2 (e.g., PR4, General ECG…). Then, those interactions will need to be examined for changes, etc.

RequisitePro actually offers the Change Management capability throughout multiple levels of traceability relationships. That is, changing a PR entry in the PRD may impact several SRs in the SRS, which may, in turn, impact several Implementation Units, which may, in turn, impact one or more Test Plans. RequisitePro also tracks the traceability linkages on a bi-directional basis. For example, changing a Test Plan specification may cause you to look back to the Implementation Units (IU) for potential impact. In turn, changing an IU may require a re-inspection of affected SRs and may even require a re-inspection of the top-level PRs, which are ultimately linked via the traceability relationships you established.

In all cases, RequisitePro tracks through the traceability links and inserts the Suspect Link markers wherever appropriate. This powerful facility provides an easy way for you to track the impact of changes in your project.

Change History Audit Trail

RequisitePro offers a powerful facility for maintaining an audit trail of changes. The most useful part of this feature is the automatic tracking of changes made to individual requirements. RequisitePro manages each and every requirement separately, regardless of the document containing the requirement. Thus, all changes you make to each requirement will be captured automatically by RequisitePro and these changes can be recalled for later inspection and review.

The change history captures the current statement of the requirement including the current values of all of the requirement's attributes. By capturing all of the current requirement parameters, you can use the history as a compact way of viewing all of the requirement's parameters. This is similar to the usual attribute views offered by other facilities in RequisitePro.

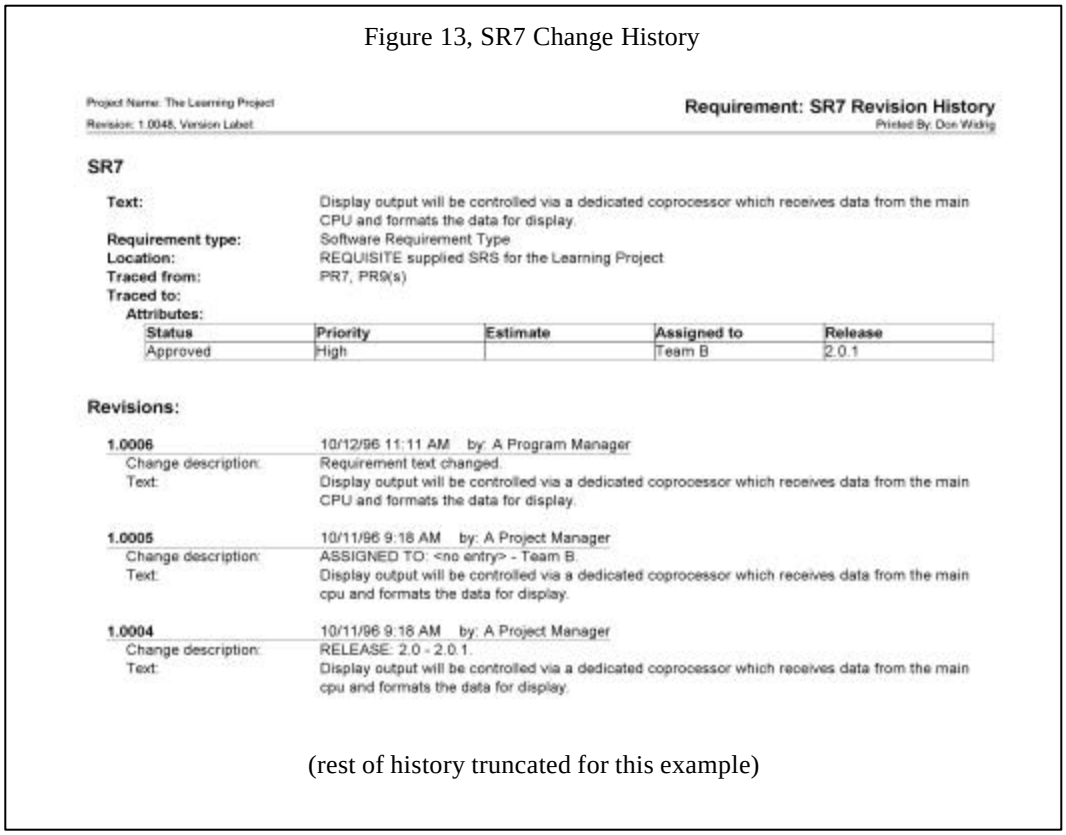
The change history also allows you to view a chronological history of all prior changes to the requirement, including its attributes. RequisitePro automatically captures all changes to the text of the requirement and changes to the values for the requirement's attributes.

Whenever RequisitePro detects a change, the background for the change is automatically captured. In addition, RequisitePro includes an automatic capture of the author of the change (i.e, the person making the change with RequisitePro) and the date and time of the change. Then, at any future time, the chronology of the change as well as the change author can be viewed as part of the history record.

In addition, RequisitePro allows you to enter a change description to document the change. Typically, you might enter a sentence or two to explain why the change was made, make references to project memos regarding the change, etc. Documenting the change will provide a satisfactory rationale and cross-reference so that later inspection of the history can adequately recall the motivation for the change. This will be a key element in FDA review of those changes that affect the clinical claims, efficacy and safety of the device.

A sample printout of a partial SRS requirement history (SR7) is shown in Figure 13. Note that the change history is arranged in reverse chronological order and records both changes to the text (change #1.0006 vs #1.0005) and changes to the values of selected attributes (change #1.0005). Text changes can be very tiny such as the change in capitalization of the word "cpu" in the example text of 1.0005 and 1.0006. Nevertheless, the minuscule changes are considered a change and are logged appropriately by RequisitePro.

Figure 13, SR7 Change History



Project Name: The Learning Project
Revision: 1.0048, Version Label:

Requirement: SR7 Revision History
Printed By: Don Widrig

SR7

Text: Display output will be controlled via a dedicated coprocessor which receives data from the main CPU and formats the data for display.
Requirement type: Software Requirement Type
Location: REQUISITE supplied SRS for the Learning Project
Traced from: PR7, PR9(s)
Traced to:
Attributes:

Status	Priority	Estimate	Assigned to	Release
Approved	High		Team B	2.0.1

Revisions:

1.0006 10/12/96 11:11 AM by: A Program Manager
Change description: Requirement text changed.
Text: Display output will be controlled via a dedicated coprocessor which receives data from the main CPU and formats the data for display.

1.0005 10/11/96 9:18 AM by: A Project Manager
Change description: ASSIGNED TO: <no entry> - Team B.
Text: Display output will be controlled via a dedicated coprocessor which receives data from the main cpu and formats the data for display.

1.0004 10/11/96 9:18 AM by: A Project Manager
Change description: RELEASE: 2.0 - 2.0.1.
Text: Display output will be controlled via a dedicated coprocessor which receives data from the main cpu and formats the data for display.

(rest of history truncated for this example)

Configuration Management and Change Management

The powerful change history feature exists at three levels within a RequisitePro project:

1. At the finest level of detail, the change history records all changes to each individual requirement within the project. This is the level of detail exhibited in Figure 13.

2. At a middle level of detail, RequisitePro, used in integration with popular industry configuration management tools, including PVCS and Visual Source Safe, automatically maintains a similar change history for each document that is known to the project.

3. At the most general level of detail, RequisitePro, used in conjunction with configuration management tools, automatically maintains a similar change history for the entire project. In this mode, RequisitePro also provides security of access to prevent unauthorized changes to crucial project documents. At the project level, RequisitePro also maintains a built-in project archiving feature to allow you to "snapshot" the project at a particular plateau of development..

With these features, RequisitePro provides an automatic and seamless integration to common applications that will assist you in the Configuration Management tasks which are critical to managing high assurance software projects.

Conclusion

With the advent of the latest FDA regulations, the medical device manufacturer is being faced with more stringent guidelines governing the processes employed in the development of medical devices and medical device software. It can be expected that this trend will continue. The price of poor design control appears not only in the failure to pass FDA review, but also is experienced in products that don't meet customer expectations, project delays that overshoot schedules by half with associated cost overruns, and in the most extreme case, termination of the project.

In parallel, we are developing increasingly complex systems that require better understanding of the components that make up the project. Rising customer demands are making a systematic approach to design control an absolute must. Understanding requirements management processes and utilizing these in the building of medical devices is the fundamental building block in a successful approach to the design, test, and management of projects.

By combining the ability to import and retrieve requirement documents in their original form and by tying this to a central repository that includes the requirements, specifications, attributes and the traceability links between them, a controlled mechanism for assuring the consistency and quality of the design is established. Through the use of RequisitePro, the project team can manage the device design process, improve team communications, define project baselines more clearly, and manage resources more efficiently. In addition, RequisitePro provides automated support for requirements traceability and change management, thus reducing development cost and improving resultant quality by eliminating many of the error prone manual activities.

Incorporating RequisitePro into a medical device team's design control process provides a more automated means to develop products that are delivered on time, within budget, that satisfy the customer's true needs, and assure patient safety.

Suggested Reading

Software Requirements - Objects, Functions, & States, Davis, Alan M., Englewood Cliffs, NJ: Prentice Hall, 1993.

Exploring Requirements - Quality Before Design, Gause, Donald C., and G. Weinberg, New York, NY Dorset House Publishing, 1989

For information on how to order these books, or for addresses of the available internet forums discussing requirements management, please contact Rational Software Corporation, 4900 Pearl East Circle, Suite 106, Boulder, CO 80301, phone (303) 444-3464, fax (303) 444-3413, e-mail: information@rational.com

Glossary of Abbreviations

510(k)	The shorthand reference to the body of governing legislation that covers the application to market medical devices which are similar to pre-existing devices already in the marketplace. Used in a manner similar to "401(k)" when referring to a federally regulated company savings plan.
CDRH	Center for Devices and Radiological Health
CGMP	Current Good Manufacturing Practices
CMP	Configuration Management Plan
FDA	Food & Drug Administration
EU	European Union
GMP	Good Manufacturing Practices
HA	Hazard Analysis
HRS	Hardware Requirement Specification
IEC	International Electrotechnical Commission
IU	Implementation Unit
IEEE	Institute of Electrical and Electronic Engineers
ISO	International Standards Organization
LOC	Level Of Concern
MDA	Medical Device Amendments
ODE	Office of Device Evaluation
OG	"Office of Device Evaluation Guidance for the Content of Premarket Submission for Medical Devices Containing Software (draft document)"
PRD	Product Requirements Document
QSR	Quality System Regulation
SMDA	Safe Medical Device Act
SRS	Software Requirement Specification
V&V	Verification and Validation
VVP	Verification and Validation Plan

Bibliography

Davis, Alan M. *Software Requirements - Objects, Functions, & States*. Englewood Cliffs, NJ: Prentice Hall, 1993.

Davis, Alan M. *201 Principles of Software Development.* New York, NY: McGraw-Hill, Inc., 1995.

FDA. *Medical Devices; Current Good Manufacturing Practice (CGMP) Final Rule; Quality System Regulation*. Washington, DC: GPO, 1997.

FDA/ODE. *ODE Guidance for the Content of Premarket Submission for Medical Devices Containing Software*. Washington, DC: GPO, (draft 1.3, 12 Aug 1996).

Gause, Donald C., and G. Weinberg. *Exploring Requirements - Quality Before Design*. New York, NY: Dorset House Publishing, 1989.

IEEE. *IEEE Standards Collection, Software Engineering.* IEEE: New York, NY. 1994.

Leffingwell, D., and A. Davis, *Using Requirements Management to Speed Delivery of Higher Quality Applications.* Rational Software TR0001, 06/96.

Wood, Bill J, and Julia W Ermes. "Applying Hazard Analysis to Medical Devices", *Medical Device & Diagnostic Industry* magazine, 01/93.