Valued Quality. Delivered.

Closing the Gap

Understanding the Software 510(k) Review Process

For more information about Intertek's Testing and Certification capabilities, please contact Intertek at 1-800-WORLDLAB, email icenter@intertek.com, or visit our website at www.intertek.com.

Introduction

The creation of quality medical software is a challenge to the medical device industry. Getting a medical device cleared through the FDA premarket 510(k) approval process can be just as challenging, especially if it has software included. The role of software in medical devices has evolved to encompass more and more functionality. Devices that were previously seen as quite rudimentary are now including software for more critical functionality. The medical device industry is slowly moving toward higher levels of features, networking, data gathering, imaging, and diagnostic functions. The complexity is growing from the most complex radiological devices all the way down to the infrared heat lamp. Software, of varying levels of complexity, can be found in all of them.

The question on the minds of many medical device manufacturers is: *How can we best ready ourselves for the premarket approval process?* The answers have been provided by the FDA through the following guidance documents:

- Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices

- General Principles of Software Validation; Final Guidance for Industry and FDA Staff

- Guidance for Industry, FDA Reviewers and Compliance on Off-The-Shelf Software Use in Medical Devices

These guidance documents are the number one resource for all 510(k) submitters as they have each been FDA-created and approved. They contain guidance that will be used when reviewing your software documentation, so it is in your best interest to follow them as closely as possible and be prepared to provide the information they are requesting whenever possible.

The First Step – Understanding

The first step in preparing 510(k) software documentation is to understand the mindset behind the FDA guidance documentation and the mindset of a potential reviewer. We can't predict how a reviewer will think, but we can understand the basic structure within which they are reviewing, and that is one of the primary aims of this document.

What is software as it relates to a medical device *and* as it relates to a 510(k) review? Software, other than the most rudimentary, is much more than just another component of a device that makes it work. Inside the software component box, you will see that the parts – while not physically moving or electrical – are all interconnected. If one of these conceptual, documental, procedural parts malfunctions there could be costly and dangerous outcomes.

The outputs and inputs are interconnected through the complex interoperation of the software. If you take timelines out of the software lifecycle you can see the process connected within itself and outward to the device manufacturing staff and engineers, and the software and hardware development environment used to create this software. It's this drilling down that will put the width and breadth of what is being reviewed into perspective.

The creator of a 510(k) that includes software – especially software that is a Major Level of Concern – should not view the software as a part of a machine, but rather as an entirely separate entity. The software is connected to this medical device by more than just inputs, functions, sub-routines, objects, classes, and outputs. It is the sum of the parts that work towards its creation – from initial device description to hazard analysis and through all requirements, the architectural layout, design specifications, verification and validation, revisions, and death.

The software review for 510(k) devices is the review of each of these parts, including how they come together and form a quality component that fits and works correctly in the design of the medical device. Essentially, the software review is actually a review within a review, and that's the best way to approach it from both a reviewer and manufacturers' perspective.

That is not to say that some medical device software isn't going to be little more than some basic assembly code that takes input, goes through a simple decision tree, and dumps out an output to a piece of hardware. This type of software may appear basic, but it is still as good as what goes into it. This applies to 510(k) software documentation as well. The submittal will only be as good as what is put into it – and this is what a reviewer must contend with.

The *Least Burdensome Approach* – Why it's Important

You may have heard of the "Least Burdensome Approach" – but what is it really? The word "burden" is generally ambiguous and subjective. Luckily, the FDA offers a definition of "burden" as it relates to the *least burdensome approach* to reviewing premarket submissions. Here it is:

> "A successful means of addressing a premarket issue that involves the most appropriate investment of time, effort, and resources on the part of industry and FDA."

The Least Burdensome Approach is part of the FDA Modernization Act of 1997. Basically, this act was an attempt to streamline what had become a burdened system of review. It was “burdened” in the way that it caused delays in getting new medical devices to market. Lead times were long and the system heavily favored big business that had the funds and resources to get their products through the FDA system. This was a poor situation for small business, doctors and patients, all of whom felt the effects of delays in new medical device technology reaching the market. Although the FDA took it upon themselves to reduce what manufacturers saw as unnecessary (burdensome) requirements – those that greatly increased a medical device’s time to market – they failed to reduce statutory requirements for proving that a new device is substantially equivalent to a previously marketed device. Therefore, the FDA Modernization act was born and the Least Burdensome Approach was brought forth.

Several years later the FDA created a guidance document for the approach: The Least Burdensome Provisions of the FDA Modernization Act of 1997: Concept and Principles; Final Guidance for FDA and Industry

The review process for 510(k)s is underpinned by the Least Burdensome Approach. It is important to understand this approach and to look at it objectively, remembering that the requirements for proving substantial equivalence have not and will not change. A submitter of a 510(k) still must show that their device is as safe and effective as a previously cleared predicate.

The methods for demonstrating substantial equivalence are the primary avenues for lessening the burden on manufacturers. To review, let’s start with what the FDA states within the document mentioned above.

The FDA says:

FDA and industry should focus on those issues that can affect the substantial equivalence (SE) determination, which describes whether or not the device has the same intended use as the predicate device and is as safe and effective as a legally marketed device. Information unrelated to the substantial equivalence decision should not be submitted to, nor requested by, the Agency.

They go on to state:

In making the SE determination, the Agency should reaffirm its longstanding review policy that:

1. *Substantial equivalence will normally be determined based on comparative device descriptions, including performance characteristics; and*

2. *Performance testing should be submitted if there are important descriptive differences between the device and other devices of the same type or the descriptive characteristics for the new device are not precise enough to assure comparability. In these instances, the most appropriate bench and/or animal testing, or in the case of IVDs, analytical testing (i.e., precision, accuracy, limit of detection, cross-reactivity, and effects of interfering substances, and clinical*

> *sensitivity/specificity), to address the performance issue should be provided. Summary information regarding the testing should generally suffice, but the test protocol, description of test methods, or any standards followed in conducting the testing should also be provided.*

Why is this important from a device manufacturer's perspective? The FDA is requesting proof that your new device is as safe and effective as the predicate, while making you aware of all items and documentation required to demonstrate that equivalence. Essentially, the FDA wants to know: *How similar is your device to the predicate? What are the differences? Will these differences make your device less safe or less effective than the predicate? If so, how can you prove it?* These are the basics. The 510(k) review process is simple, and the FDA does a wonderful job of reiterating these basic statements time and time again in text, checklist, and chart form – each time leading back to the basic premise of understanding the differences, similarities, and proof of safety and effectiveness for each device.

The next logical question is: *How does this relate to software found in a medical device?* The answer follows logically as well. If the Least Burdensome Approach is being used to submit and review your 510(k) submission, it will also be used to evaluate the software information you've provided. What is less clear, however, is how to prove that your software is as safe and effective as the predicate. If you're comparing your new device to your company's previously marketed device, and the software is almost identical, then this becomes a relatively easy process. However, most device manufacturers don't have this luxury, and this is where the FDA Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices comes into play.

This guidance documentation is what the FDA thinks is the Least Burdensome Approach to showing that a new device's software is as safe and effective as the predicate device's software. The safety and effectiveness is relying on a quality system in place that will create a structural system that will naturally maintain the concept of good in - good out. This substructure of a quality system will maintain quality processes that surround and support your software lifecycle, and thus support the premise that the new device's software is as safe and effective as the predicate. Essentially, the information requested in the software guidance documents should be readily available to a medical device manufacturer following a standard of engineering that meets certain quality systems.

The FDA suggests that manufacturers of software devices should create and maintain software-related documentation in accordance with the requirements of the Quality System Regulation (QS Regulation; 21 CFR Part 820). This provides a strong overview of what the FDA is looking for from a Current Good Manufacturing Process (CGMP) standpoint, without going into a software-specific quality system. The FDA does have consensus standards for software lifecycles (i.e. AAMI SW68), but there are many different types of software lifecycles available that meet both the needs of the individual device manufacturer and the requirements of a quality system. The absence of a software quality system and lifecycle process leads to some difficulty for the medical device manufacturer when managing the 510(k) submission.

Having a complete quality (hardware and software) system in place up front will be a valuable ally for the device manufacturer because it provides a level playing field for both parties. You now have a common language to speak based on the quality system you are using. In this way, you can close the gap between what documentation a manufacturer may have and what the guidance (and the reviewer) is requesting.

Where to Start

Not all devices have software as a major component of their device. In fact, some medical device manufacturers think so little of their software that they forget to mention it in their 510(k) submission. For example, your software could be nothing more than a little timing chip programmed to turn something on or off. Does this example apply? From an FDA standpoint the answer is: “it depends.” A better question to ask is *What is the software controlling?* This approach stems from a hazard based analysis of the device. Correctly conducting a hazard analysis or risk assessment of your device will answer these types of questions for you. If your risk assessment determines that the software is not controlling anything that would relate to the safety and effectiveness of the device, then a simple discussion of why it does not impact the safety or effectiveness of the device – in relation to the legally marketed predicate – may suffice. Please note that this is a relatively rare exception, rather than a rule.

There are many reasons to include software functionality in a medical device. It may be a necessary part of the overall function of the device, or it may make a device more efficient, smaller, safer or more effective. It may even just be included to make the device more flexible or simple. It is best to approach this as information that will help the 510(k) reviewer to understand whether the device is substantially equivalent to the previously marketed predicate, even if such information is not technically required.

The guidance document attempts to separate the levels of information needed based on the software’s Level of Concern. The problem with this approach is that, while the FDA tries to be flexible with what information they think is necessary as part of the Least Burdensome Approach, it neglects to give medical device manufacturers a strict baseline for software guidance.

As a rule of thumb, if the information is suggested by the guidance document and it directly correlates with determining the safety and effectiveness of the new device so as to be compared with the predicate, it would be best to include it. This is not to say that you should throw volumes of unnecessary documentation into a box. The best practice is to keep the information relevant to the guidance document and the determination of substantial equivalence (i.e. the Least Burdensome Approach).

The FDA Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices is naturally focused more on the information related to Moderate and Major Level of Concern software since these are more hazardous, but this doesn't let the manufacturers of Minor Level of Concern software devices off the hook. Table 3 within this guidance document outlines the basics required for Minor Level of Concern devices, but additional information that it is relevant to the substantial equivalence determination should always be provided. These Minor Level of Concern devices may not require an entire lifecycle process, but the FDA guidance asks for documentation that provides the basis of a software lifecycle process.

Some important things to remember:

- Follow the Least Burdensome Approach.
- If the information is relevant to showing substantial equivalence, provide it.
- Streamline the process. It saves time to provide information up front, rather than have it requested during the review.
- A software lifecycle process may not be required, but it makes things much easier.
- Substantial equivalence has to be shown and substantiated in terms of safety and effectiveness to the predicate. The quality of the software is a key part of this determination.

General Guidance for Devices Containing Software

1. Determining the Level of Concern

An incorrect determination of the Level of Concern for a device's software can be a costly mistake. The gray area between Moderate and Major Levels of Concern lies within the definition of "serious injury." The definition of death is very straightforward, but the term "serious injury" is more subjective.

A "serious injury" is defined by the FDA in 21 CFR 803.3(bb)(1) as an injury or illness that:

1. *is life threatening;*
2. *results in permanent impairment of a body function or permanent damage to a body structure; or*
3. *necessitates medical or surgical intervention to preclude permanent impairment of a body function or permanent damage to a body structure.*

If cases where this definition is still not clear, manufacturers may be forced to self-assess the Level of Concern for their device. The FDA may or may not agree with this self-assessment, and a disagreement can lead to delays. Therefore, if you are uncertain about

Level of Concern, the FDA suggests that you contact them directly to discuss your device software. By putting in this extra effort up front, you can eliminate lengthy delays during the review process. If you are interested in consulting the Agency for informal guidance, then you may need to participate in the Pre-IDE Program.

Ideally you want a submission to provide enough information so that questions aren't asked. Following the guidance and providing ample information for whatever Level of Concern your software may be will give you the advantage of covering the obvious questions that a reviewer may face.

Once you've determined what Level of Concern your software is, document it. The software review lives or dies at the hands of documentation. The FDA wants to see how you came to determine your Level of Concern. The process works best when hard facts or data are used to make the point. Assertions without substantiating data are much more questionable and increase the likelihood of delays.

2. Software Documentation - Overview

As mentioned earlier, the FDA differentiates the amount of documentation needed in the software review based on the software Level of Concern. This is a logical way to approach the problem of making sure to cover all of their bases, but it tends to leave some room for interpretation. The Minor Level of Concern software device manufacturers can immediately see from Table 3 what they *don't* have to submit – but that's only half the answer in terms of what information should be included. The FDA has built a level of flexibility in the guidance that takes into account the many different types of devices that may use software and therefore need this guidance. This flexibility lacks some firm boundaries or baselines to provide medical device manufacturers with firm validation that they are saying enough or including too little.

First and foremost we want to focus on the mission of the 510(k) process, which is to show substantial equivalence to a previously marketed device as it relates specifically to safety and effectiveness. While this information should be contained within your software documentation, too much information is going to hinder the process and too little will cause delays. Some reviewers may look for more information while others may look for less. The baseline is meant to ideally bridge the gap and make sure that you have a sense of what may be expected of you. Knowing the expectations will help you provide as much information as necessary to make the review process move smoothly.

Ultimately, guidance documents for specific device types and software should be followed, and any areas in question should be answered fully with detailed information and hard data. If there are further questions that may need to be answered, the device manufacturer should seek advice from the FDA via a Pre-IDE process.

3. Software Description

All Levels of Concern should have a software description. The software description should arise from the complexity of your device and its software. If you touch on all the points mentioned in the guidance summary of what you should have in your software description, you will have the bare minimum. And while the bare minimum may be okay for the simplest software, higher Levels of Concern require more information to provide a full description of the software. This approach to writing a software description is no different from the approach for writing your overall device description. Essentially, your job in preparing the 510(k) software description is to fully educate the reviewer on your software.

When describing the overall device in the device description of the 510(k), most manufacturers start by describing the device's function, then describe each functional part as it relates to the system as a whole. The description may also identify the components used in the construction of the device. Submitters may include schematics, diagrams, or photos to aid in this description, which will generally:

1. Explain, in detail, what the device is doing
2. Support your claim of substantial equivalence by describing how the device is functioning in comparison to how the predicate device
3. Support the SE claim in the area of safety by illustrating the safe functioning and use of safe components in the construction of the device

The software description should try to illustrate these same concepts, addressing the following types of questions in paragraph form:

- What purpose is this software fulfilling?
- What is the software doing and controlling in reference to the device?
- Is it being used to control some hardware or display information?
- Is the software monitoring or calculating or displaying anything?
- What kind of significant features is it offering?
- Is it controlling or monitoring functions that are detrimental to the safe operation of the device?

The description should be broad enough to answer these questions without delving too deeply into the inner workings of the software. The functional description should then lead into a description of how the software is functioning in a very basic sense. The FDA is looking for information on the operational environment. This is where you talk about the nuts and bolts of the software: *How was it written? What is it running on (hardware and OS)? Is there "Off the Shelf" software being used?*

These questions require more than a one word response. Your answers should be based on hard information. For example, when answering what programming language was used, elaborate on the specific type and version of programming language and indicate what type of compiler was used. If you're software is using an operating system, it is best to note the specific version.

In determining whether all this information is necessary, consider the specific issues that may arise based on the device and the software complexity, as well as the Level of Concern. If the Level of Concern is minor, then a one word answer, although not ideal, will most likely suffice. If the Level of Concern is moderate or major, specifics should be given. Pinning down specifics such as the hardware platform or programming language is one of the simplest ways to avoid review delays.

4. Device Hazard Analysis

A Device Hazard Analysis is suggested for all levels of concern. From the most minor to the most major, there should be some analysis of the risks involved. The Hazard Analysis is a somewhat touchy subject, which goes back to having a quality system in place and a solid software lifecycle.

The Hazard Analysis is a very concrete document that deals with very specific information. If it is not available, then it needs to be created. The Hazard Analysis should be a natural extension of your software design process. The complexity of the analysis is going to be directly correlated to the complexity of the device and the software itself. This should be reflected in the Risk Assessment (additional information on Risk Assessment and Management can be found in the "Additional Topics" section of the software guidance document). The FDA references ISO 14971 specifically, but there are many Risk Assessment methods available. Ideally, you should be following a concrete method which has been documented and is readily available and based on substantiated and standardized methods.

The Hazard Analysis, in the FDA's eyes, is a lynchpin of the review process. While manufacturers may use many different software lifecycle processes, documentation methods and risk assessment ideologies, the Hazard Analysis is key in showing that the 510(k) subject device is as safe as the predicates. The Hazard Analysis should be firmly integrated into all other parts of the software documentation. This is a key concept called traceability.

The guidance document contains a document called the Traceability Analysis. Traceability highlights the state of the quality system that is being used to create your device and, specifically, your software. For very complex devices and software, traceability is the key to making sure that all parts of the design process are correlated together and verifiable. Traceability is like the skeleton running through the body of information that is produced as part of a very complex device. It would be nearly impossible to confirm all the parts of a software design at that level of complexity. It is important that the manufacturers of Moderate and Minor Level of Concern software understand the concepts so that the information is provided and delays are avoided. The FDA is asking for very specific information. This information locks a medical device manufacturer into a more specific level of quality management system and software lifecycle.

The Hazard Analysis has most likely identified a number of hazards with associated mitigations to ensure a certain level of safety. Traceability is the best method of making sure that these hazard mitigations are being integrated into the software design and validation process. It is an underlying sense of organization that radiates throughout the software documentation to enlighten the reviewer.

For example, you may have a hazard that you have assessed as being a relatively high risk of injury to a patient. This kind of hazard should have some way of monitoring it to ensure that the mitigations for this type of hazard are included in the design of the device.

This method is based on the assumption that the medical device manufacturer is performing their Risk Assessment early in the design cycle. Risk Assessments are often done at each stage of design so as to include any new risks that may have been identified during the course of the design process. These risks are then folded back into previous design stages as necessary and included moving forward.

All hazards should be identified in some way. There are different ways to track a hazard, and tracking via specific number schemes is common. In this example, we will reference hazard number 13, or “HAZ13.”

> **HAZ13:** The software in the device has the potential to blind the patient if a failure allows the patient to get unregulated output. The mitigation of this hazard is by the use of software limitations to the output and monitoring of the output.

We will continue this example in the sections to follow.

5. Software Requirements Specifications (SRS)

Software requirements are usually an extension of the overall device requirements that are created early in the software lifecycle process. Device requirements can vary greatly in their level of specificity. Software requirements could be as vague as stating, “The movement is controlled by software.” From a hardware perspective this is acceptable, but more detailed software requirements will need to be written.

Software requirements should be more than just a list of what the software does. A good quality system will ensure that the requirements cover at least a bare minimum and a basic level of detail. A summary of the requirements is adequate for Minor Level of Concern devices. However, it is to your advantage to touch on the requirement aspects outlined in the guidance, thereby reducing the number of questions that may arise during the review.

The best software requirements are built in a logical way. They begin with very basic requirements outside of what the software will actually do, and describe physically how the software exists. If hardware requirements deal directly with how the software will execute and integrate into the design of the device, then these correlate with the software requirements. This is a factor that is explicitly stated in the FDA guidance document, but is often overlooked by device manufacturers. The list included in the guidance isn’t exhaustive, but is a great place to start. This documentation may be dictated by the quality and lifecycle processes put into place, but the 510(k) software requirements specification should be able to answer the following:

- Where will the software physically be located (memory)?
- How will it execute (microprocessors)?
- How will it physically interface with the rest of the device (input / output hardware)
- How is it powered?
- What kind of safety hardware is involved?

Safety cannot be stressed enough. Identifying all aspects of the software requirements that include safety related requirements will go a long way in helping a reviewer understand how this device is being engineered to be as safe as the predicate.

The software requirements specifications can be broad, but nevertheless should provide distinct traceability to other aspects of the software design, such as the Hazard Analysis and the software design specifications.

Programming and interface requirements are generally very straightforward, but may be glossed over in some software requirements specifications. This will depend on the specific device and its implementation.

Software performance and functional requirements usually take up the bulk of the SRS documentation. This is also where the gradual and logical building of the software requirements can be useful. The flow of the requirements, following the guidance, bridges from hardware to software and details the interaction of the two. The process can begin anew once the software performance and functional requirements begin. Manufacturers often build details based on the architecture of the software (detailed later in this document). The architecture can range from very basic to very detailed and is highly dependent on the complexity of the device and the software. However, maintaining a sense of logical order throughout the documentation is a valuable way to ensure traceability.

Hazard Analyses can be performed in varying logical ways, including subsystem, hazard type and architectures. This varies because most risk management evaluations include an entire device (both software and hardware). Exceptions usually arise when the software component is very complex or makes up a large part of the device functionality.

Software requirements, however, are purely software related and can logically follow the same architectural structure or object oriented structure that is built into a software system. That is not to say that this is the only way or best way to organize performance and functional requirements. This is often the most sensible and least burdensome approach that will, ideally, catch all requirements as they are broken into functional parts. Organization should be established based on the quality system, the device and other specific factors that go into software lifecycle processes. That being said, the requirements for functionality and performance should cover the information outlined in the FDA guidance regardless of how it's organized. The requirements should cover all aspects of the software functionality, even if the requirements seem somewhat broad.

Requirements that arise from the software Hazard Analysis are generally seen within the functional and performance requirements. It is a good practice to provide some type of traceability within the SRS, at least to hazard based requirements. The Traceability Analysis should account for full traceability between hazards, requirements, verification and validation activities, etc. Adding some built-in traceability, specifically for hazards, works to reinforce the traceability throughout the manufacturer's documentation and aids in the reviewer's understanding of how hazard mitigations are being actively implemented.

Continuing our example from the last section, we identified HAZ13 which dealt with hazardous output that is being mitigated through software. Examples of software requirements stemming from this hazard mitigation are:

SRS52: The software must monitor device output using a light sensor (HAZ13)
SRS53: The software must engage light limitations based on the feedback from the light sensor (HAZ13)
SRS54: The software must be able to safely filter or extinguish the light output (HAZ13)

There may be more requirements that stem from just the one hazard mitigation and there may also be requirements that are interrelated to these requirements, such as a sensor requirement or a requirement for the actual light limitation values. These should all be traced back to requirements that directly or indirectly insure that the device is correctly mitigating the hazard found in the hazard analysis. The Traceability Analysis should show all of these interrelations, but it often helps to have the traceability built into the specific documentation as a further method of organization.

6. Architecture Design Chart

The software architecture design is very helpful in graphically explaining what the software is doing, how the pieces fit together, and how they interact with the hardware. The architecture design chart deals primarily with Moderate and Major Level of Concern devices and can be used as a road map of the software development and documentation process.

This section discusses design charts specifically, but it is often necessary to include charts, state diagrams, and discussion to convey a full understanding of how the pieces of the software puzzle fit together. As stated in the guidance, the design chart should show the relationships among the major functional units in the software, starting with basic structure and then breaking down those basic structural blocks into more functional units. This is especially true for Major Level of Concern devices or highly complex software. The reviewer needs to understand how the software code is conceptually structured, then how these structures are receiving input from the outside world. Additionally, they need to know how the functional parts are working together and transferring data. It may be necessary to further describe how each individual part is processing data or provide state diagrams. Finally, they will need to know how software data is being sent back out of the software and into the hardware world.

The detail level will depend on complexity, functionality and Level of Concern, but generally a high level of detail is not necessary if the functionality is very simple and not directly related to the safety and effectiveness of the device. On the other hand, if the functional block is complex or related to safety and effectiveness, then the design flowcharts (including state diagrams) should have a level of detail to enable a reviewer to understand the full relationship and functionality. This is especially true for Major Level of Concern devices. A discussion of the general architecture or specific flowchart or state diagram – while not specified in the guidance – can also assist the reviewer in fully understanding the software architecture, especially if the flowcharts contain vague or ambiguous elements that need more detailed information. This may seem like more effort than is suggested, but making sure that the software architecture is well understood can help reduce delays caused by additional reviewer questions.

7. Software Design Specification (SDS)

Detailed design specifications and traceability are going to make or break any Moderate or Major Level of Concern software review. The SDS is also not necessary for Minor Level of Concern devices, but this does not mean that it is not an important and necessary part of a well-managed and documented software lifecycle.

The SDS is based on the medical device manufacturer's previous Software Requirements Specification. It is the roadmap to success and the logical next step in implementing quality software. The SDS essentially allows the software developer to show that the requirements have been broken down into their constituent parts. This detailed breakdown of the requirements shows how the software pieces will be implemented, while providing backwards traceability to the software requirements that they are based upon. This document is also very important in demonstrating to the FDA reviewer that the design and implementation are thorough, and the work completed was detailed, calculated and controlled based on a software lifecycle process. A well-documented SDS also carries the manufacturer forward to a well-designed and thorough Verification and Validation plan.

It is important to note that the SDS generally mimics the SRS documentation in format, owing to the fact that the SDS forms a natural extension of the initial software requirements. However this may not always be the case, which is why traceability between the documentation is so important. For very complex systems, the device manufacturer can break the documentation up into specific sub-system requirements and design specifications. If you have a solid basis of traceability that can be shown in a Traceability Analysis, then the requirements and design specification can be easily traced no matter the size or complexity of the system.

Continuing our previous example, we had three sample software requirements: SRS52, SRS53, SRS54 (see page 12). These were each traced back to the hazard identified previously, HAZ13 (see page 10). The Software Design Specifications may mimic the software requirements in their numbering system, but it is often not an option for a device with a high level of detail. There are a number of methods for resolving these types of problems, but they aren't always necessary if the traceability is well documented.

For this example we will break down SRS53 into a few design specifications with a numbering scheme based on the SRS numbering:

> **SDS53.1:** The software will monitor feedback via a Brand X light sensor with the following specifications.
> **SDS53.2:** The following light levels will be adhered to based on software hard limits.
> **SDS53.3:** The light sensor will be calibrated in the factory.

This is a very simplified illustration of how Software Design Specifications can be broken out. However, it is easy to see the inherent traceability built into the documentation through the use of a logical numbering scheme. The SDS numbers are based on their corresponding SRS numbers with extensions to break the requirement down to its base implementable parts. This will be important in the Traceability Analysis and also in demonstrating to the reviewer that there is an internal cohesion in the design process that flows throughout each part of the software lifecycle.

8. Traceability Analysis

Traceability is necessary for all Levels of Concern. This can be one of the easiest or hardest documents to create, depending on the software development process that has been followed thus far.

The Minor Level of Concern manufacturer – if following a well implemented software lifecycle – should have all of the information at their disposal that a Major Level of Concern manufacturer would have. At this point it is just a matter of creating a table to tie everything together. If the software review documentation is a body, then the traceability is akin to the central nervous system touching all parts of the body. It is an often impossible task to reverse engineer a nervous system into the body of software documentation you may create, which is why it is so important to follow a well documented software lifecycle process that includes traceability from the first step to the final.

When providing a Traceability Analysis, we find that a table format is often the most direct and simple way to illustrate how the parts of the software documentation are tied together. Remember to account for all applicable software hazards, requirements, design specifications, and verification and validation (specifically important for Moderate and especially Major Level of Concern devices).

The following is an example of how our internal examples would flesh out to a Traceability Analysis.

Hazards	Software Requirements	Software Description	Verification and Validation
HAZ13	SRS52	SDS52.1	INT_T03, SYS_T33
		SDS52.2	INT_T03
	SRS53	SDS53.1	SYS_T08, SW_T27
		SDS53.2	SYS_T09, SW_T27
		SDS53.3	--
	SRS54	SDS54.1	HW_T08, SW_T12
		SDS54.2	SW_T58, SW_T59
		SDS54.3	SW_T58, SW_T59
		SDS54.4	SW_T58, SW_T59
--	SRS55	SDS55.1	SYS_T97
--	SRS56	SDS56.1	SYS_T12

As you can see, this excerpt shows that our hazard example, HAZ13, can be traced to software requirements. These requirements are traced to Software Design Specifications, and all of these are traced to a number of different verification and validation tests. It is important to note that not all requirements must be linked to specific hazards, such as SRS55 and SRS56, which may be general requirements not directly associated with a hazard. It's also worth pointing out that SDS5.3 is actually a requirement that is related to manufacturer calibration and would not have an associated software verification and validation test.

The level of detail provided in the traceability analysis will be a direct result of the complexity of the medical device software, the lifecycle processes, and the internal manufacturing processes used to create the medical software. We have seen Traceability Analyses range in length from one page to several hundred. It is important to stress that this is the most common method used to demonstrate traceability, but it is far from the only method. The manufacturer is urged to use the method that is least burdensome and most appropriate for his software complexity, lifecycle and quality systems.

9. Software Development Environment Description

This is another area that is only necessary for Moderate and Major Level of Concern devices. This is the 510(k) submitter's opportunity to give the reviewer an overview of their software lifecycle processes. This documentation should give the reviewer a real look into the process you are using to create your software, but it should also delve into the nuts and bolts of the system in place to control the software lifecycle as well.

Many manufacturers feel that a diagram of a software lifecycle is sufficient within this section of their submission, yet the reviewer generally expects to see a detailed paragraph description of how you've created your software. This should include an overview of the lifecycle, a description of the in-place processes used to manage each lifecycle stage or activity, and any changes that may occur during development.

Configuration and change management is a very important aspect of the software development environment. The FDA is very interested in how you go about making and tracking design changes at every stage of the software lifecycle, including the often forgotten post-market lifecycle stage. This description should include the control of requirements and specifications changes, test plans, and regression testing as necessary to ensure that any software changes have been well documented and tested.

The environment description should include a summary of the configuration management and maintenance plans at least for Moderate Level of Concern devices. These should include pre and post release maintenance. Major Level of Concern devices should include a very thorough description of the configuration management and maintenance for all stages of the lifecycle. These often include the discussion of the internal tracking programs used to track development changes, bug fixes, and aftermarket improvements or corrections. Major Level of Concern devices will also need to include an annotated list of the control/baseline documents that were generated as part of the software development process and a list or description of software coding standards.

Providing a thorough description of the environment in which the software is developed will give the reviewer a solid framework upon which to base his review, and provides assurance that there is a solid process in place to handle design changes as they may occur. The FDA has included Software Change Management in the "Additional Topics" section at the end of the software guidance for a more thorough discussion of the subject.

10. Verification and Validation Documentation

Verification and Validation (V&V) Documentation is the key to demonstrating that the culmination of the software development process has created a stable, safe and effective medical device software. It is through verification and validation that the manufacturer can demonstrate that the software requirements and design specifications has been completed correctly. It also shows that the software outputs meet the input requirements for each phase of development as necessary based on the complexity of the device. Verification activities are not just testing, although the terminology is often used interchangeably. Verification may include the following from the software guidance document:

- Code and documentation reviews
- Walk-through Analysis
- Various Static and Dynamic Analyses
- Module Testing
- Integration Testing
- System Testing

Validation, on the other hand, demonstrates that the software specifically meets the user needs and the intended function for the device. Verification and Validation activities often overlap and go hand-in-hand.

V&V Documentation is necessary for all Levels of Concern. The amount of information provided is going to depend on the complexity of the device software, the lifecycle process followed, and the Level of Concern.

For Minor Level of Concern devices, the FDA suggests system or device level testing and, where appropriate, integration testing. Integration testing is often overlooked by submitters and should be provided if possible. This demonstrates that the device works correctly and that all hazards, if any, were mitigated properly. This type of testing should integrate into the level of Software Requirements Specification information provided previously for Minor Level of Concern devices.

All tests submitted should include the pass/fail criteria and a summary of the test results. Information is generally presented in a table or other list format and contains a brief description of how the test can pass or fail and a result or a summary of the end-results.

The Moderate Level of Concern suggestions build on what is necessary for the Minor Level. The Agency states that the submitter should include a summary list of all validation and verification activities and the results of these activities. This includes the pass/fail criteria. The Agency is now asking for all of the information from the Minor Level of Concern (system, device, and integration testing), as well as, any other verification or validation processes or tests that may have been conducted.

This may include code reviews, walkthroughs, testing for the integration of software modules into a more complete design, and the system or device level testing. The overall level of information is still much the same as with the Minor Level of Concern devices. You will need to provide the test identification number for traceability, as brief description of what is being tested, and how it can pass or fail, and the result.

Major Level of Concern builds again on the Moderate Level. For Major Level of Concern devices the manufacturer will include all information previously outlined for the other Levels of Concern, as well as unit testing that may have taken place prior to integration testing and during the software development cycle. All test failures should be noted and described, in addition to any modifications or retesting that may have taken place. Any software bugs found would be fleshed out in the “Unresolved Anomalies” section of the submission (See Section 12 below).

Overall, the verification and validation section of the software submission should be a cinch for manufacturers who follow a well-defined software lifecycle process and internal quality system. The information requested should be on hand and it should just be a matter of tailoring the level of detail based on the Level of Concern for the device. Of course, this may not be the case for all manufacturers so it is important to remember that employing a solid lifecycle early can make all the difference between no delay and long delays at this late stage in the game.

11. Revision Level History
This is relatively straightforward for all levels of concern. The guidance specifies that a list of all major software revisions should be provided up to and including the current software version. This is often a line item list of dates, changes, and new version numbers. It is important here to include all changes, especially if they impact the safety or effectiveness of the device, in order to accurately track the progress of the software. The last entry, as stated before, should be the final version of software that will be in the marketed device.

There have been instances where the final tested version of software differs from the final marketed software version. In this case, the revision level history should include any changes between the two versions as well. These untested changes between the final tested version and final marketed version can create additional questions for reviewers. It is always better to provide as much information and reasoning for any untested additions and changes as possible to avoid any confusion or misinterpretation when the final marketed software differs from the final tested software. This should include an assessment of the potential impact of the differences in the final marketed and tested software versions as they relate to safety and effectiveness.

12. Unresolved Anomalies
This is a list of any leftover bugs that were not resolved during the initial design and testing for the device. The list is necessary for all Moderate and Major Level of Concern devices and should include:

- A description of the bug or anomaly
- How this uncorrected software anomaly impacts the functionality of the device with special attention on how it could impact the overall safety and effectiveness of the device
- Any mitigations or workarounds you may have for these unresolved bugs
- A timeframe or description of how and when the problem will be corrected.
- If, and how, you will be communicating these bugs to the product end-user

The Agency is very interested in what kinds of software bugs may be leftover after the pre-market software development process is completed. They understand that all software defects may not be able to be corrected based on the size of the issue, project timelines, and other factors. However, it is important to document these outlying anomalies and how they could impact the device in any way. The guidance document also notes specifically that the impact of these bugs should be analyzed as they relate to safety, effectiveness, operator usage and human factors issues.

13. A Note on Off-the-Shelf Software
Off-the-Shelf (OTS) software may be one of the most misunderstood sections of the software submission. OTS software is often overlooked by device manufacturers and can lead to many questions and delays.

If your device is using any software that was not expressly written by or for your company you need to submit information on for the Off-the-Shelf software as per the Guidance for Industry, FDA Reviewers, and Compliance on Off-The-Shelf Software Used in Medical Devices. This includes, but is not limited to, the following:

- Operating systems used on the device
- Networking software
- Database software
- Software that may be used on-board the device in conjunction with microprocessors
- Viewing software for images taken by the device
- Bluetooth or other wireless software

Review timeframes can be drastically impacted by not including information as per the Off-the-Shelf guidance document for software that is being used in the operation of your medical device. The guidance is well written, easy to follow, and quite straightforward as to what the Agency is seeking about OTS software.

About Intertek

Intertek is a leading provider of quality and safety solutions serving a wide range of industries around the world. From auditing and inspection, to testing, quality assurance and certification, Intertek people are dedicated to adding value to customers' products and processes, supporting their success in the global marketplace. Intertek has the expertise, resources and global reach to support its customers through its network of more than 1,000 laboratories and offices and over 30,000 people in more than 100 countries around the world. Intertek Group plc (ITRK) is listed on the London Stock Exchange in the FTSE 100 index.

For more information about the 510(k) approval process, the FDA Third-Party Review Program, or specific software requirements, contact Intertek today.

icenter@intertek.com
1-800-WORLDLAB