www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

Improve Your Ability to Sell Software into Medical Device Companies

Introduction

It is estimated that the medical device business will reach nearly 105.8 billion in the U.S. for 2011. If you have a software application which can be sold to medical device manufactures, you stand to benefit. However, as the market is highly regulated, it can be tricky getting your product past the front door. In this paper we will share five things you can do to make your software application more attractive to medical device companies.

Concepts to Improve Medical Device Adoption

1. Demonstrate your software application is developed and maintained with mature software development practices
2. Leverage international and adjacent industry standards to demonstrate quality
3. Partition your software application to minimize risk for your customer
4. Form partnerships with related vendors/tools and offer tested solutions
5. Provide a Computer System Validation (CSV) starter pack to accelerate adoption of your software application

How do you demonstrate to a medical device company that you have a mature SDLC model?

Why would they care? Let's start with some background information.

All medical device companies doing business in the U.S. are subject to FDA regulation. In 1989, the FDA issued a policy statement regarding software. They wrote that "oversight of software should depend primarily on the risk to the patient should the software fail to perform to its specifications."

And, as software complexity continued to increase during the 90's, the FDA determined that additional guidance documents needed to be issued to keep pace. Some of these included:

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

The FDA continues to make incremental updates on these and other software related areas.

These guidance documents really exist to help medical device manufactures comply with the overarching 21 CFR 820 Quality System Regulations (QSR). As there are many areas covered in this regulation, I'll mention just a few of the key ones here.

The QSR is broken out into the various development phases. It covers design controls, including input and output requirements, and design review and verification. It also includes production and process control. It includes document controls as well as change control.

The QSR governs all the areas in a lifecycle process from the design phase (where the manufacturer determines the user specifications) to the process controls and then to the operation and maintenance of the software. Devices will be considered "adulterated" by the FDA if a manufacturer does not meet the requirements of the QSR.

The QSR also covers what manufacturers should do, and procedures manufacturers should have in place, to handle nonconforming products or "customer complaints." These are known as *corrective and preventive actions* (CAPA). Read our past SPK white papers on CAPA.

There is quite a bit more detail about 21 CFR 820 on several websites. So now that you have some of the background information and resources, what can you do to show medical device companies that your *software application is developed with good practices*?

Is there a certification process for 21 CFR 820? Unfortunately No. Is there enough information in the FDA Regulations and Guidances to just use it verbatim to do your software development? No again. The truth is that the FDA does not specify the methodology that a business must use to develop software. So where do you go from here? One way is to leverage the international standard, IEC 62304, the Medical Device Lifecycle Process.

The FDA's CDRH organization, which is the device center, has acknowledged the advantages of international standards at "reducing the documentation requirements on manufacturers." Conformance with IEC 62304 provides evidence of having a software development process in place, and fulfills the requirements of 21 CFR 820 (as well as the Medical Device Directive 93/42/EEC).

In other words, if you are building your software application leveraging IEC 62304, medical device companies will quickly understand and feel more confident about your development model. You're talking the same language as many of them, and you will set your product apart from other competitors.

A few of the key areas covered within 62304 include:

- The Software Development Process

  Planning, Requirement Analysis, Architecture Design, Detail Design, Unit test implementation and verification, Integration testing and System Testing and then Software Release.

- The Software Maintenance Process

  Planning, Modification Analysis & Implementation

- Software Risk Management

Builds on IEC 14971 (which is general risk management) and adds specific software classifications.

- Software Configuration Management

  Management of access, and version control of software artifacts

Seriously consider sharing with your perspective medical device manufacturer/customer all your documentation on your SDLC processes, validation & verification test cases, and evidence (data) of test competition.

Leverage international & adjacent industry standards to demonstrate quality

We already spoke about IEC 62304. There are other standards that may be useful in demonstrating quality in your software product.

As an example, let's take a look at IEC 61508-3, Functional Safety of Electrical Electronic Programmable Electronic Safety-related Systems (software subsection). This requires that:

"Software safety functions and software systematic capability are specified; establishes requirements for safety lifecycle phases and activities which shall be applied during the design and development of the safety-related software."

The interesting part is that you can attain an actual certification for IEC 61508. It is based on receiving an SIL (Safety Integrity Level) rating indicating the probability of failure for low demand or continuous running of your application. For example a rating of SIL1 is the lowest level (highest risk). A rating of SIL4 is the highest level (least risk). This helps demonstrate to your perspective medical device customer an additional commitment to safety using established risk management techniques.

Another approach is to leverage certifications you already may have in other industries. For example, if your company sells software to the avionics manufacturers, you are probably already certified for:

- DO-278, Guidelines for Communications, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance.

  It uses assurance levels (AL1-AL6) to describe the risk impact of your software (in tandem with DO-178B –Airborne Systems). AL1 – Catastrophic, to AL6 – No Effect.

  The DO-278 guidelines are in the form of:

  - Objectives of software life cycle processes
  - Description of activities and design considerations for achieving these objectives
  - Description of the evidence that indicate that the objectives have been satisfied

  And

- DO-178B, Software Considerations in Airborne Systems and Equipment Certification

- It uses software levels (A-E) to define the failure level. Level A – Catastrophic, to Level – E No Effect.

The DO-178B guidelines are in the form of:

- Software Planning Process
- Software Development Process

- Correctness, Confidence & Control Process

Do you see a pattern here? You'll find many of the same software development best practices are duplicated in one form or another, from industry to industry. Use them to your advantage in promoting your safety and risk management maturity.

Partition your software application to minimize risk for your customer

Let's revisit IEC 62304 once again. IEC 62304 indicates the manufacturer must classify their software system (as a whole) for risk into 3 classes:

- Class A: No injury or damage to health is possible
- Class B: Non-serious injury is possible
- Class C: Death or serious injury is possible

Items in Class C require more activities than B or C.

Table I: Summary of safety classification effects on the code development documentation and process.

Software Documentation	Class A	Class B	Class C
Software development plan	Must contain contents to sections 5.1 IEC 62304:2006. The plan's content list increases as the class increases, but a plan is required for all classes.		
Software requirements specification	Software requirements specification conforming to 5.2 IEC 62304:2006. The content list for the software requirements specification increases as the class increases, but a document is required for all classes.		
Software architecture	Not required.	Software architecture to 5.3 IEC 62304:2006. Refined to software unit level for Class C.	
Software detailed design	Not required.	Document detailed design for software units. (5.4).	
Software unit implementation	All units are implemented, documented and source controlled (5.5.1).		
Software unit verification	Not required.	Define process, tests and acceptance criteria (5.5.2, 5.5.3). Carry out verification (5.5.5)	Define additional tests and acceptance criteria (5.5.2, 5.5.3, 5.5.4). Carry out verification (5.5.5).
Software integration and integration testing	Not required.	Integration testing to 5.6 IEC 62304:2006.	
Software system testing	Not required.	System testing to 5.7 IEC 62304:2006.	
Software release	Document the version of the software product that is being released (5.8.4).	List of remaining software anomalies, annotated with an explanation of the impact on safety or effectiveness, including operator usage and human factors.	

Source – European Medical Device Technology, June 2010, Volume 1, No. 6 written by Ken Hall

So therefore, if you are selling software into a medical device company that will be a component or accessory to a device, it is advantageous to:

1) Allow the vendor separate your high risk software from your low risk software.

   For example, embedded OS vendors often have the ability to separate critical software from the rest of their system using virtualization techniques. They try to make the critical portion as small as possible. This helps potentially to reduce the amount of Class C software.

2) Enable the vendor to take the minimal amount of software required. Less software may equate to less risk, and therefore less required testing. I.e. turn off unwanted features.

   For example, embedded OS software vendors often can shut off/remove parts of the kernel that is not needed.

The benefits to your customer of partitioning your software also include improved maintenance (fewer areas to retest with an update).

Of course, your mileage/approach will vary depending on the software architecture of your product. In some cases it may be impossible or impractical offer this type of customization.

Form partnerships with related vendors/tools and offer tested solutions

If your software application works with another vendor, you may wish to reach out to that vendor to create a combined validated solution.

For example, if your customer is using a hardware platform, they will need the interface software between the operating system and the hardware device, known as a BSP (Board Support Package). If you and your hardware vendor partner to create and offer a certified validated solution, it will increase your customer's safety and quality confidence with your application.

Provide a Computer System Validation (CSV) starter pack to accelerate adoption of your software application

Your customers will experience an improved adoption rate of your product with a perceived lower barrier to internal deployment if you provide them with a validation kit.

The FDA via 21 CFR Part 820(i), specifies the following for validation of automated processes:

> *"When computers or automated data processing systems are used as part of production or the quality system, the manufacturer shall **validate computer software for its intended use** according to an established protocol. All software changes shall be validated before approval and issuance. These validation activities and results shall be documented."*

This validation kit should include a Risk Assessment, a Validation Plan, Business & Functional Requirements, IQ/OQ/PQ Protocols and Output Summary Documentation. All items should be tied together via a Traceability Matrix. It should also cover CFR Part 11 (Electronic Signature) items in addition to the core usage of the tool by the customer (the SOPs).

Without a validation kit, the customer often takes weeks to build all the data from scratch. That alone has sometimes presented enough of an obstacle to slow down sales.

SPK and Associates are engineering consultants with expertise in the medical device industry. Call us today to help create a validation package for your business or any discuss any of the topics covered here.

Carlos Almeida
SPK and Associates
Architect, Software Engineering