Effective Software Verification for Medical Devices

Achieving compliance and meeting productivity goals with static analysis

In addition to producing some of the most complex code with the fewest software engineering resources relative to other embedded programming industries, today’s medical device manufacturers must also satisfy a stringent regulatory environment. New regulations dictate that manufacturers must follow specific best practices when developing software for medical systems. These regulations differ from country to country and are periodically introduced, reviewed and updated – making it difficult for manufacturers to stay abreast of all necessary rules.

This paper will clarify the current regulatory requirements for medical device software and outline the compliance and productivity benefits of automated static source code analysis (SCA) technology. The paper will also discuss software verification best practices that device manufacturers can incorporate into their development processes to help achieve compliance and productivity goals while developing reliable, secure and feature-rich software code.

The Regulatory Environment – A Brief Overview

Both the European Community (through European Community directives) and the U.S. Food and Drug Administration (FDA) have developed regulations that must be satisfied before a medical device can be classified and approved for use. In addition to specifying requirements for the marketing, labeling and monitoring of medical devices, these organizations also require that device manufacturers validate the software components embedded in their products.

FDA Requirements

Medical devices sold within the United States are subject to both the general controls of the U.S. Federal Food, Drug and Cosmetic (FD&C) Act and the final procedural regulations in the Title 21 Code of Federal Regulations Part 800-1200 (21 CFR Parts 800 – 1299). In particular, 21 CFR Part 820 provides standards for setting company policies, operating procedures, guidelines and objectives to institute a quality system framework.

Moreover, the FDA has produced a General Principles of Software Validation document that “…outlines general validation principles that the Food and Drug Administration (FDA) considers to be applicable to the validation of medical device software or the validation of software used to design, develop, or manufacture medical devices.”

These principles apply to:

- Software used as a component, part, or accessory of a medical device
- Software that is itself a medical device (e.g., blood establishment software)
- Software used in the production of a device (e.g., programmable logic controllers in manufacturing equipment)

1 General Principles of Software Validation, FDA, January 11, 2002, Section 1.
http://www.fda.gov/RegulatoryInformation/Guidances/ucm126954.htm
» Software used in implementation of the device manufacturer’s quality system (e.g., software that records and maintains the device history record)

Lastly, the level of testing – and the amount of test documentation required to be submitted for a project – is specified in the FDA Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices document. These requirements differ by the software’s “Level of Concern” which can be classified as Major, Moderate, or Minor according to patient or operator risk.

In assessing whether a medical device is compliant, the FDA looks for objective evidence that a structured process was used in the development of the device’s software. It therefore recommends that companies use a layered software validation approach that includes software verification. The FDA notes that, “Planning, verification, testing, traceability, configuration management, and many other aspects of good software engineering … are important activities that together help to support a final conclusion that software is validated.”

In its assessments of medical devices, the FDA also looks for evidence that the design outputs of a particular phase of the software development life cycle have met all of the specified requirements for that phase. The FDA examines whether software testing and verification activities – including various static and dynamic analyses, code inspections, document inspections, and walkthroughs – have been used to confirm that the software development output has met its input requirements.

In addition, the FDA requires that all software validation and verification work must be carried out in compliance with the Quality System Regulation outlined in 21 CFR Part 820. The FDA Quality System Regulation (QSR) provides manufacturers of finished medical devices with a framework of basic requirements to use in the establishment of a quality management system. As with the requirements laid out in its General Principles of Software Validation document, the QSR requires that the verification activities produce evidence that the development processes – and resulting product – meet stated requirements.

**International Requirements**

Medical device manufacturers must also be aware of the directives issued by other regulatory bodies such as the European Community (EC). While each government authority has produced its own compliance directives, there is a good deal of overlap between them.

Similar in scope to the FDA’s Quality System Regulation, the ISO 13485 standard requires that manufacturers demonstrate that a software development quality system has been implemented and maintained. Furthermore, the EN 62304 standard (issued by CEN (European Committee for Standardization), CENELEC (European Committee for Electrotechnical Standardization) and ETSI (European Telecommunications Standards Institute)) requires that activities similar to those outlined in the FDA’s General Principles of Software Validation document be conducted and documented according to the software’s Class. These Software Classes are very similar to those used by the FDA as shown (in simplified manner) in the following table:

<table>
<thead>
<tr>
<th>RISK TO THE PATIENT OR OPERATOR</th>
<th>FAILURE UNLIKELY TO CAUSE ANY INJURY</th>
<th>FAILURE COULD RESULT IN MINOR INJURY</th>
<th>FAILURE COULD RESULT IN DEATH OR SERIOUS INJURY</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDA Level of Concern Guidance</td>
<td>Minor</td>
<td>Moderate</td>
<td>Major</td>
</tr>
<tr>
<td>EN 62304 Classification</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

While the global regulatory environment for medical device software may appear complex, all regulatory bodies are essentially seeking the same proof points:

2 http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm089543.htm

1. Evidence that the development of the software took place within a quality system framework.
2. Evidence that the software has been verified and validated.

To ensure that an appropriate quality system is in place, medical device companies should consult the ISO 13485 guidelines and consider engaging an independent, third-party expert to assess their quality system to identify any areas of concern. When it comes time to verify and validate a device’s software, manufacturers have a number of techniques and technological options from which to choose.

**Software Verification & Validation**

Proper software verification and validation techniques should be key components of any medical device compliance strategy. The FDA considers software validation to be “confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled.”

Software validation activities:

» Should occur at every stage of the software development life cycle
» Should demonstrate that all software requirements have been implemented correctly and completely and are traceable to system requirements

No single technique can solve all validation challenges, but a combination of approaches and technologies can help medical device manufacturers produce the highest quality software possible. In addition to validating the software, organizations must verify that it has been constructed properly. The difference between validation and verification can be summarized as follows:

» Validation asks “Are you building the right thing?”
» Verification asks “Are you building the thing right?”

To “build the right thing,” organizations need to examine the specifications from the user’s point of view and ensure that the product meets the needs of its external customers. To “build it right,” manufacturers need to look internally to verify that all specifications are implemented correctly and that inputs and outputs comply with previously determined regulations or standards.

To perform software verification, organizations can choose from a variety of static techniques that include manual peer code reviews, the application of coding standards, source code analysis for uncovering programming errors, as well as a variety of dynamic analysis technologies such as unit testing and memory profiling. Dynamic techniques, while effective and necessary, are often limited in their efficiency and scalability. Dynamic techniques must execute or run the code, and this can become a costly endeavour. Companies can also encounter scalability limits when attempting to build the test cases required to achieve code coverage goals. Occasionally, for large code bases, tools like memory profilers can run for days in order to perform their analysis. This is one reason why many organizations supplement their dynamic analysis strategies with a variety of static techniques. This combination will ensure that issues that can be found more cost-effectively with static analysis are removed from the code prior to dynamic analysis, reducing time and effort.

Due to the array of available static verification techniques, many organizations find it challenging to determine the right mix of tools and processes required. For example, while peer code reviews are commonly used to ensure that only high quality software makes its way into a medical device, the scheduling and overhead associated with these reviews makes them an expensive proposition. Still, they must be a core part of most verification strategies so many organizations use a tool-supported approach in which reviews can be conducted remotely. In this case, participants are provided with a rich code inspection interface through which they can conduct their reviews.

Even if the reviews are supported by tools in a collaborative environment, there will always be an upper limit to how much code can be reviewed manually. The FDA has recognized that manual reviews are not sufficient on their own: “The complexity of even a small software application can be too much for a human-based deterministic failure analysis.”

Aside from the scalability limitations, manual code reviews can also be quite costly since they typically take senior development resources away from other tasks, which is why the reviews need to be focused on design and requirement issues.

**Source Code Analysis**

To address these challenges, many medical device companies are turning to static source code analysis (SCA) tools to help automate the code inspection process. SCA tools can detect and identify structural deficiencies such as incorrect pointer

---


5 [www.msse.umn.edu/system/files/Brian09.ppt_Slide 5](www.msse.umn.edu/system/files/Brian09.ppt_Slide 5)
usage, overflows and leaks that can cause field failures. This is particularly useful in larger projects, where developers must typically write much of the code before they have a suitably integrated system that can be executed on the target device. Moreover, static analysis tools can locate software “bugs” early in the software development process – usually long before integration builds are available for execution. The earlier bugs are found, the faster and cheaper it is to correct them – making it far more likely that they will be addressed.

The best SCA tools complement other software verification tools by delivering sophisticated analysis directly to developers, allowing them to identify and remediate coding vulnerabilities as they are developing the software. Using this approach, fewer defects make it into the code stream, leading to higher quality code, greater efficiencies, as well as more effective peer code reviews and testing.

**Compliance Benefits**

Static analysis can be an effective tool for providing evidence of compliance to regulatory bodies during the software development process, and for uncovering the root cause of a medical device issue once the system is in use. While these organizations do not prescribe specific tools and processes, they do expect that companies incorporate best practices when developing software for medical systems.

Integrating the use of static analysis into the development process can help medical device manufacturers demonstrate that they are following best practices by using a rigorous, proven tool to validate and verify their code. In addition, the use of automated tools frees developers to focus on other issues related to compliance (such as meeting design specifications) and reduces time spent on simple bug fixes. By putting developer time to productive use, medical device manufacturers can help to reduce the costs associated with achieving compliance.

Regulatory bodies also look for evidence that organizations are focused on preventing the introduction of defects into the software development process. They do not look kindly upon manufacturers who try to “test quality into” the software code after it is written. Static source code analysis technology allows development teams to find a wide range of defects prior to testing and builds confidence that the software is fit for its intended use. Moreover, static analysis tools can help medical device companies develop code that is more reliable and of higher quality – increasing the likelihood that the end product will function as intended and comply with appropriate regulations.

**Productivity Benefits**

By stabilizing code early in the software development process, developers can be freed from the constant stream of issues that otherwise would be reported downstream. For example, the use of static analysis makes peer reviews more productive by ensuring that reviewers focus on critical design and requirements issues rather than simple bug fixes and coding standard compliance. Combined with a collaborative, tool-supported code review process, organizations can streamline this part of their development process while meeting their reliability goals.

In addition to making individual developers more productive, static analysis can improve the reliability of medical devices. By eliminating coding errors that might not otherwise be caught, or by catching problems at an earlier stage, the risk of product failure in the design stage or in the field can be reduced.

SCA can also improve time-to-market for medical devices. Its use eliminates bugs early in the development process, when defects are not costly to fix, and can potentially decrease development delays along with any delays in FDA or other regulatory approvals due to software errors. Customer studies have shown measurably improved product quality at various stages of the software life when SCA was made a standard part of the development process.
In addition, many commercial SCA tools provide more than just bug and security vulnerability detection. SCA vendors that offer collaborative code review, software metrics, coding standards, and architecture analysis capabilities can help organizations reduce the number of tools required to achieve their productivity, compliance, and software quality goals. These tools also provide organizations with code that is easier to maintain and update. Clean code allows for faster product update and extension testing, which can reduce costs significantly throughout the product’s lifecycle.

Conclusion

In addition to operating in a competitive marketplace with increasing end-user demands for features and usability, medical device manufacturers operate in a highly regulated environment. Regulatory bodies look for evidence that medical devices are developed under a structured, quality-oriented development process. By following software validation and verification best practices, manufacturers can not only increase the likelihood that they will meet their compliance goals, they can also enhance developer productivity.

One of the most important verification tools that medical device manufacturers can deploy is source code analysis technology. Source code analysis tools provide an automated method to detect a significant number of software bugs or security vulnerabilities early in the development process and before any code is delivered to the testing team.

Klocwork® provides medical device manufacturers with a comprehensive range of static source code analysis and collaborative code review products that enable accurate and timely software code verification. By allowing device manufacturers to move high-quality source code analysis to the developer’s desktop and perform it at the earliest point in the development cycle, Klocwork’s SCA products help manufacturers perform the software verification and validation activities required by regulatory bodies in a timely, productive fashion.

In addition to verifying and validating software, medical device manufacturers need to ensure that their products are developed under an appropriate quality system. SterlingTech Software offers development and independent verification and validation services to medical device manufacturers that include product testing, Quality Systems assessments and GMP Process Validation. SterlingTech uses Klocwork tools as an integral part of its software validation process. By engaging SterlingTech Software, medical device companies around the world are achieving compliance success and reducing the risk of regulatory rejection.

About Klocwork

Klocwork® helps developers create more secure and reliable software. Our tools analyze source code on-the-fly, simplify peer code reviews and extend the life of complex software. Over 1000 customers, including the biggest brands in the mobile device, consumer electronics, medical technologies, telecom, automotive, military and aerospace sectors, have made Klocwork part of their software development process. Tens of thousands of software developers, architects and development managers rely on our tools everyday to improve their productivity while creating better software.

About SterlingTech Software

Founded in 1998, SterlingTech Software (STS) is an ISO 13485:2003 registered provider of full life-cycle software development, testing and validation services exclusively for the medical device industry. Medical device manufacturers engage SterlingTech for independent verification and validation of their medical software development processes – including Quality Systems assessments and GMP Process Validation. The company’s work has resulted in successful FDA 510(k), PMA and CE submissions for numerous medical device projects. STS offers a free consultation that includes a detailed software analysis and project plan for any medical device. Contact SterlingTech Software at www.sterlingtechsoftware.com or info@sterlingtechsoftware.com.