WHITE PAPER

DEVELOPING HIGH-PERFORMING MEDICAL DEVICES SOFTWARE – UNDERSTANDING WHAT LIES BEHIND IT

Infosys®
Navigate your next

Contents

Introduction 4

Medical devices software classification 4

Some standards used in the medical domain 5

Focus is on software - defective software emerging as a key reason in medical device recalls 5

Typical components used in medical device software applications 5

Key design considerations for medical device software development 6

- Design with regulatory classification in mind 6
- Design for software reliability 6
- Design for modularity 6
- Robust data structure design 6
- Design to abstract 6
- Design for secure software 6
  - Design for Code level security 6
  - Design for network security 6
- UI design to minimize user error and aid the user to provide correct inputs 7

Software traceability and software documentation. 7

Software validation 8

- Unit testing 8
- Unit test automation 8
- System testing 8
- System test automation 8

Summary 8

Introduction

Software is now an integral part of medical devices. In fact, we now have Software as a Medical Device (SaMD) in addition to being present in medical devices (SiMD). Developing medical device software demands rigor and is subjected to regulatory approvals too. Defective software leads to high costs for the device manufacturers and even device recalls.

This document, aimed at medical application developers and medical product designers, presents the factors that must be considered while developing software for medical devices subject to regulatory approvals.

It addresses architectural, design and system test considerations and shares best practices. The document also touches upon the UI design aspects as well as the traceability of the artifacts generated during the software development process. The views are derived from the author's experience developing Class 3 medical device software.

This document does not address the following aspects.

- Aspects of documentation for regulatory submissions.
- Any specific software development life cycle models
- Any specific commercial tools/ operating systems used during medical software development.
- Quality management process or medical standards followed in medical device software development. The reader can refer to Section 4.0 for more details on the applicable standards.

Medical devices software classification

Medical device software mandates special software development considerations to ensure safety.

There are three software safety classes as defined by IEC 62304

- **Class A:** The software cannot cause any harm
- **Class B:** The software can cause minor harm such as injuries
- **Class C:** The software can cause major harm, such as severe injuries or even death

There are three safety classes as defined by FDA based on their risks and the regulatory controls necessary[8]

- **Class 1:** Lowest risk that presents the minimal potential for harm
- **Class 2:** Moderate risk - higher risk than Class 1. These require premarket notification.
- **Class 3:** These devices sustain or support life, are implanted or present potential unreasonable risk of illness or injury, and require premarket approvals

Some standards used in the medical domain

The section gives an overview of the most used standards in the medical device domain.[1,2]

ISO Standards

- **ISO 13485 -** Medical devices — quality management systems — requirements for regulatory purposes
- **ISO 14971 -** Medical devices — application of risk management to medical devices
- **ISO 12207** - Systems and software engineering — software life cycle processes
- **ISO/IEC/IEEE 15288 -** Systems and software engineering — System life cycle processes

US-FDA

- **21 CFR 820 –** Quality System Regulation
- **21 CFR Part 814 -** Premarket Approval (PMA)
- **21 CFR Part 11 -** Electronic Records; Electronic Signatures - Scope and Application

> Software Concerns remained the top cause FOR recalls 21 times in the last 22 quarters. These 41 events accounted for 17.4% of third QUARTER RECALLS.

Figure 1 Medical device recalls - an overview

Focus is on software - defective software emerging as a key reason in medical device recalls

As per the Sedgwick Q3 Recall Index, medical device recalls for Q3 2021 increased nearly 36%. Figure 1 provides the details. This edition of the Sedgwick Brand Protection Recall Index focuses on U.S. product recall data and regulatory developments.[4]

Typical components used in medical device software applications

Most medical device software has the following key components - Operating System (OS)

- **OS or Platform API –** may include API for network programming, API for concurrent programming, File System API, Math API
- **Tools or libraries used in development –** may include data structure implementation, parsers
- **Test infrastructure –** may be created by the medical device software provider or by using the platform/language provided standard test infrastructure (NUnit, JUnit)
- **UI –** HTML, JavaScript etc.
- Medical domain application software

These components will need to be assessed to be classified as non-medical software, Class 1, Class 2, or Class 3.

Key design considerations for medical device software development

Since software remains a critical component, it is necessary to design the software based on the following key principles.

Design with regulatory classification in mind

Partition s software into functional components. Then, based on the functionality provided by these components, they can receive different regulatory classifications.

Design for software reliability

Using standard design and architectural patterns or creating proprietary design patterns can help reduce software. Reusing these components in a family of products ensures that these patterns are well tested, decreases the risks of errors in the software and improves the system reliability.

Software reviews will be more efficient as the review teams will understand the design or architectural patterns better.

Design for modularity

The software should be designed to be modular, comprising different components. Components should be designed to be updated separately as needed and reused across a family of products. The typical method is to use them as software libraries/binaries. In addition, components designed across a family of products bring down development costs.

Robust data structure design

The software should ensure that the application data only has relevant values. Best practices for this are:

- Define data types with an upper limit and a lower limit or a set of values relevant to the application.
- Define data types to have an initial value
- If a particular data value is dependent on another data value, then enforce the inter-dependency.
- Define invalid state as yet another part of the data value set.
- Use circular buffers to store large amounts of in-memory application data.

Design to abstract

This is a key architectural concept used in medical software development. We recommend adopting a layered architecture, with each layer performing a specific function or a set of functions

Some of the examples that could be followed are

- Communication Layer – Responsible for data communication with the external system
- Protocol Exchange Layer – Responsible for translating the data exchange protocols transmitted across systems
- View layer – Responsible for presentation to the external user of the system
- Software Update Layer -Responsible for secure software updates
- Save Context Layer -Responsible for saving the user session context and is useful in understanding user settings and triage of any issues
- Business Logic Layer – Responsible for implementing the business logic specific to a product.

Design for secure software

It is important to design the software such that the IP of the medical device vendor is well-protected and ensure that the software is not tampered with to compromise the medical device. In addition, if the medical device software connects to an external network, cybersecurity aspects will need to be considered.

Design for Code level security

Medical device software that runs in untrusted environments such as customer devices should incorporate application shielding. Application shielding includes application hardening and anti-tampering. While application hardening techniques include code obfuscation and encryption. Anti-tampering techniques include debugging detection or emulation.[3]

Design for network security

If the medical device software is connected to the network or the cloud, it is important to consider cyber security. Some of the good practices to follow are -

- Unauthorized medical device access over the network – The medical device communication can be designed such that the device will connect only to a known Edge gateway. The details of the communication link to the Edge gateway can be provided over an out of band close range wireless communication link like NFC.
- Data transfer security over the network – A secure data transfer strategy must be adopted such that data encryption is done both at the network and application layers. An out of band communication link, such as NFC, inductive telemetry must be used to transfer the encryption keys. Wireless standard bodies, e.g., Bluetooth SIG in the Bluetooth Core Specification, recommend encryption keys of at least seven octets.[8]
- Data integrity checks when sent over the network - Data should always be encapsulated into

packets with a cyclic redundancy check (CRC). The receiver should then evaluate the CRC for validity.

- Malicious attacks on the medical device - The medical device could be subject to malicious attacks. One way to mitigate an attack is for the medical device to discard the data when it finds the received data is invalid when opening a connection. The termination of the connection could be done based on repeated receipt of invalid data.
- Vulnerabilities in cybersecurity may present a risk to the safe operation of networked medical devices using OTS software. A well-defined mechanism should be in place to apply the OTS software vendor provided software patches in a timely manner.[7]

UI design to minimize user error and aid the user to provide correct inputs

Some of the good practices are -

- Design the UI such that it is not confusing to the user.
- Design the UI to prevent an action that the user cannot perform.
- Design UI to allow only permitted values for data to be entered by the user.
- Understand the environment in which the software is to be used and the workflow that a medical practitioner follows when using the software before designing the UI.
- Harness existing popular paradigms of UI design. For example, touch and swipe gestures prevalent in smartphones.
- Design the UI to provide sufficient messages to guide the correct data input intuitively.
- Design the UI to present key application errors or warnings that impact patient health.

Software traceability and software documentation.

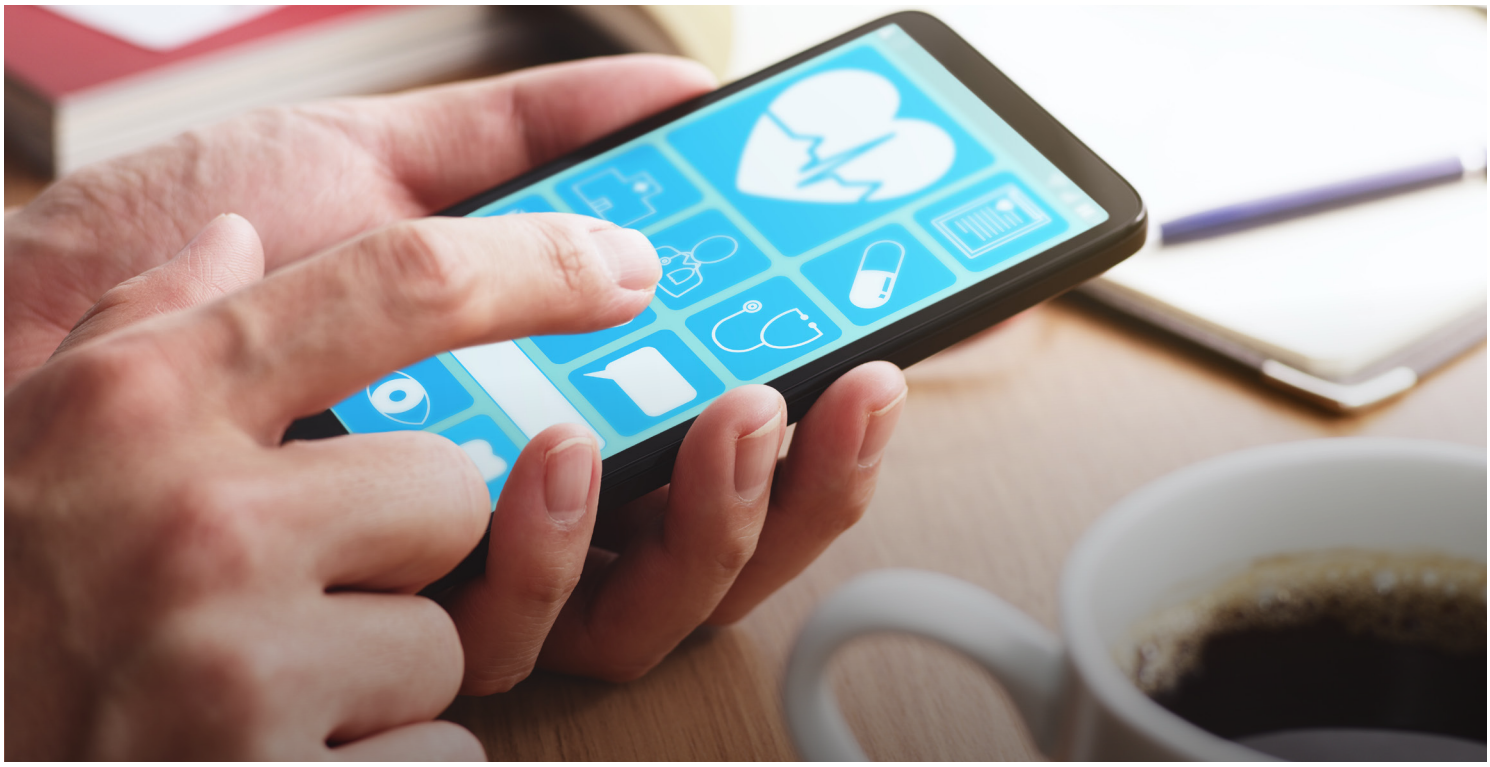
Software traceability and code level documentation are critical components for medical device software development.

Medical device manufacturers typically use tools covering all aspects of the software development process, including source code management and integrating the source code developed with the unit tests performed. The source code developed should be traceable to the design inputs or the software requirements. Likewise, verification or system tests should have traceability to design inputs or software requirements.

Defect tracking tools will need to be used to track and trace them to the software fixes applied to resolve them.

There should be a specific repository to cover field issues and track their resolution.

Code level documentation assumes significance; further, complex software implementations and algorithms should be described in detail in the code



Software validation

In medical device software development, the length of the validation phase is significant and could account for as much as 40% of the software development lifecycle.

Since the software can come from different sources, the validation aspects should cover the software from all the sources. The resultant software validation process should be commensurate with the safety risk associated with these software components. Regardless of the source of components, the software developer and device manufacturer retain the responsibility for valid software.[6]

- In-house developed software: This component is completely under the control of the medical device software vendor and can be managed better. Validation best practices discussed in this paper can be adopted.
- Off-the-Shelf Software [OTS] (compilers/debuggers/operating systems/tools): It is good to maintain formal business relationships with the OTS software vendors, thus ensuring timely receipt of information about quality problems and recommended corrective and preventive actions.[5]

Unit testing

Unit Testing plays a key role in medical device software development. During development, software tools should help in code coverage and static analysis. Therefore, unit tests should, at a minimum, follow these guidelines.

- Statement coverage: All statements in the code will need to be exercised by the unit test cases.
- Multi-condition coverage: All possible conditions in the software's decision making must be exercised by the unit test cases. Typically, this is achieved through "Truth Tables."
- Loop coverage: Unit tests should cover all aspects of the execution of the program loop from zero iteration to maximum iteration. Unit tests should check for scenarios where the software may not exit the loop resulting in a critical software issue.

Unit test automation

Unit test automation wide use in medical device software development has become a norm. It can incrementally verify that existing functionality is not broken as software features are added.

- It is good to use unit test frameworks specific to the language in which the software is developed, e.g., NUnit for C#.
- Many medical device manufacturers invest in building custom unit test frameworks that help them significantly reduce the cost and time of software development.

System testing

As most medical device software is developed for a family of devices, software maintenance can represent a very large percentage of the total cost of software over the product life cycle. As a result, an established comprehensive software validation process can reduce the long-term cost of software by reducing the cost of validation for each subsequent release of the software. In addition, many established medical device vendors are automating at least 98% of system automation tests.

System test automation

Existing automation tools that can automate user actions on the UI, such as touch and other gestures, must be utilized. These test tools also compare the resulting screen transitions and verify whether the transitions are as expected, thus significantly reducing manual testing.

Automation tools can also help monitor the system's CPU usage and memory while executing the tests and detect any memory issues that the software may be causing.

Summary

Some of the best practices addressed in the document are -

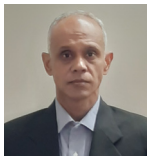
- Design with regulatory classification in mind.
- Design for reliability using standard design and architectural patterns or create proprietary design patterns.
- Medical device software that runs in untrusted environments such as customer devices should incorporate application shielding.
- Enforce application data validity through the data structure design.
- If the medical device software is connected to the network or cloud, it is important to consider cybersecurity aspects such as data encryption, CRC checks and authentication of network connections.
- All complex software implementation should be documented well in the code.
- Develop automated unit and system level tests

Medical device software can transform medical products and drive innovation in medical devices. But medical software must be safe and reliable. It can be achieved through careful design of software and comprehensive validation of the software, using best practices followed by industry leaders in this domain.

References:

1. https://www.fda.gov/regulatory-information
2. https://www.iso.org/standards.html
3. https://digital.ai/application-protection
4. https://www.sedgwick.com (RECALL INDEX 2021 Edition 3. Product Recall United States Edition)
5. https://www.fda.gov/media/71794/download (Guidance for Industry, FDA Reviewers and Compliance on Off-the-Shelf Software Use in Medical Devices)
6. https://www.fda.gov/media/73141/download (General Principles of Software Validation; Final Guidance for Industry and FDA Staff)
7. https://www.fda.gov/media/72154/download (Guidance for Industry -Cybersecurity for Networked Medical Devices Containing Off the-Shelf (OTS) Software)
8. https://www.bluetooth.com/specifications/bluetooth-core-specification/

Author



Balaji Lakshmi Narasimhan

Senior Principal Technology Architect, Infosys

For more information, contact askus@infosys.com

Navigate your next