

# IEC 62304 Audit Report

## Executive Summary

*A few paragraphs summarizing the major IEC 62304 compliance gaps and how to fill them. Usually, filling them means adjusting the software development processes. We usually suggest making one big process change at a time (e.g., one per sprint). This allows the team to adapt to the new process and collect feedback before making the next process change.*

## Methodology

This IEC 62304 audit is based on:

1. Interviews with software engineering management regarding their current and desired software processes
2. Design history file documents
3. QMS documents
4. The source repositories and project management tools

After collecting this information, we categorized compliance with each section of IEC 62304 in the "Status" column using these values:

- "System" means QMS-level procedures fulfill the section
- "Unnecessary" means we don't need to comply with this section
- "Complete" means the section is already being complied with
- "Partial" means significant progress has been made toward complying with the section
- "Incomplete" means we're not complying with the section

Although we did our best to be complete, some sections may be marked Incomplete, Partial, or Complete due to an oversight on our part.

We explain how we arrived at the compliance status in the "Status justification" column. The final column suggests how the gap can be filled, including specific suggestions for your situation.

Some rows have additional information that can be found by opening the sub-page (click the "Open" button when hovering over that row). There are also a few rows that have more detailed proposals (e.g., 5.1.7).

To fully understand the table, you'll likely need a copy of IEC 62304 (we recommend printing a copy) to read along side the sections.

## Terms

**Formal requirements** describe *what* to build without describing *how* to build it. Each requirement should be uniquely identifiable (e.g., using an id like REQ-32), expressed in terms that avoid ambiguity, not contradict other requirements, and be stated in terms that permit the establishment of test criteria and the performance of such tests. Here are a few examples:

- Users shall be able to begin a new exam in less than 10 seconds after the power button is pressed.
- The date, time, and exam number shall be recorded in a log message when a new exam is started.
- The user interface shall prompt the user before deleting an exam.

Someone who fills the **product owner** role is able to answer requirements questions. They know *what* the team is building even if they don't know how to build it. The software engineers, on the other hand, typically don't fully know *what* needs to be built. The product owner also doesn't fully know what needs to be built; they don't fully know the user's needs. Thus, formal

requirements can't be completely defined up-front. One purpose of sprint planning meetings is to have the product owner and the engineers meet to review and revise the requirements.

**Verification** is making sure all the formal requirements are met. **Validation** is making sure the device meets the user's needs. Thus, validation is checking that the product owner did their job while verification is checking that the engineers did their job.

Each project has one **project manager**. They are responsible for managing progress of the project and ensuring development is performed according to the appropriate regulations.

A **regulatory release** is a version of the software, and corresponding documents, that is either submitted to regulatory bodies for review and/or is used in clinical devices.

Three terms identify the software decomposition. The top level is the **software system**. The lowest level that is not further decomposed is the **software unit**. All levels of composition, including the top and bottom levels, can be called **software items**. A software system, then, is composed of one or more software items, and each software item is composed of one or more software units or decomposable software items. The responsibility is left to the manufacturer to provide the definition and granularity of the software items and software units.

#### IEC 62304 Gaps







Aa Section	☰ Category	☰ Description	⬇ Status	☰ Status justification	☰ How to fill the gap
<a href="#">4.1</a>	General	Quality Management System (E.g., by following ISO13485 or CFR 820)	Out of Scope	A full QMS audit is out of scope, although we note the IEC 62304 standard requires a full QMS.	
<a href="#">4.2</a>	General	System Risk Management	Out of Scope	A full ISO 14971 Risk Management audit is out of scope, although there are a few risk management activities done by the software team which are discussed below. See the notes in 5.1.7 in particular for one strategy to have the software risk management and full device risk management interact.	
<a href="#">4.3</a>	General	Software Safety Classification	Pending		
<a href="#">5.1.1</a>	Planning	Software development plan	Pending		We can help draft a software plan that compiles with IEC 62304 and also fits your desired development procedures. We can provide templates we've developed on previous projects that use an agile development methodology. We'll also need to talk with RA/QA to coordinate how the software development procedures, design controls, and document controls interface with the QMS and the system-level design.
<a href="#">5.1.2</a>	Planning	Updating software plan	Pending		See suggestions.

Aa Section	☰ Category	☰ Description	⌵ Status	☰ Status justification	☰ How to fill the gap
<a href="#">5.1.3</a>	Planning	References to system design and development	Pending		
<a href="#">5.1.4</a>	Planning	Standards, methods, and tools	Pending		Add some instructions about required standards and tooling to the software plan.
<a href="#">5.1.6</a>	Planning	Integration and testing planning	Pending		
<a href="#">5.1.7</a>	Planning	Software risk management planning	Pending		We can help draft the software risk plan. See my suggestions regarding the overall approach.
<a href="#">5.1.8</a>	Planning	Software documentation planning	Pending		We suggest that these documents be listed in the software plan.
<a href="#">5.1.9</a>	Planning	Software configuration management planning	Pending		Basically, we need a plan to show how we can reproducibly build the software and control versions of our source code, our tooling, any third-party libraries we use, and documentation.
<a href="#">5.1.10</a>	Planning	Supporting items to be controlled	Pending		See note at 5.1.9
<a href="#">5.1.11</a>	Planning	Control before verification	Pending		We need to archive copies of tools and third-party libraries prior to verification testing. We also need a long-term place to store released software artifacts.
<a href="#">5.2.1</a>	Requirements	Document formal requirements; tie to system requirements	Pending		See suggestions.
<a href="#">5.2.2</a>	Requirements	Content of requirements	Pending		We can help write or review an initial set of software requirements once we've come up with a process (see suggestions in 5.2.1). We can also help train engineers on how to write good formal requirements.
<a href="#">5.2.3</a>	Requirements	Risk control measures as requirements	Pending		
<a href="#">5.2.4</a>	Requirements	Re-evaluate risk analysis upon requirements changes	Pending		See my suggestions in 5.1.7. This re-evaluation can be most easily checked during PR reviews.
<a href="#">5.2.5</a>	Requirements	Re-evaluate system requirements	Pending		Need to discuss once the QMS has settled a bit.
<a href="#">5.2.6</a>	Requirements	Verify software requirements	Pending		See suggestions.

Aa Section	☰ Category	☰ Description	⌵ Status	☰ Status justification	☰ How to fill the gap
<a href="#">5.3.1</a>	Design	Document architecture	Pending		See suggestions.
<a href="#">5.3.2</a>	Design	Document interfaces	Pending		
<a href="#">5.3.3</a>	Design	Document functional and performance requirements for SOUP	Pending		I can provide some guidance regarding how to write these.
<a href="#">5.3.4</a>	Design	Document hardware and software requirements for SOUP	Pending		
<a href="#">5.3.5</a>	Design	Identify segregation necessary for risk control	Pending		
<a href="#">5.3.6</a>	Design	Verify the design	Pending		See suggestions.
<a href="#">5.4.1</a>	Design	Software decomposition	Pending		I can provide some guidance regarding how granular to make the decomposition. I can also provide some points regarding how to create these diagrams and can review the diagrams produced by the team.
<a href="#">5.4.2</a>	Design	Detailed design	Pending		
<a href="#">5.4.3</a>	Design	Detailed design of interfaces	Pending		
<a href="#">5.4.4</a>	Design	Verify detailed designs	Pending		See suggestions in 5.3.6
<a href="#">5.5.1</a>	Code and Test	Implement software units	Pending		
<a href="#">5.5.2</a>	Code and Test	Establish software verification process	Pending		
<a href="#">5.5.3</a>	Code and Test	Software unit acceptance criteria	Pending		See suggestions.
<a href="#">5.5.4</a>	Code and Test	Additional software unit acceptance criteria	Pending		
<a href="#">5.5.5</a>	Code and Test	Perform software unit verification	Pending		
<a href="#">5.6.1</a>	Integration	Integrate software units	Pending		

Aa Section	☰ Category	☰ Description	⌵ Status	☰ Status justification	☰ How to fill the gap
<a href="#">5.6.2</a>	Integration	Verify software integration	Pending		
<a href="#">5.6.3</a>	Integration	Test integrated software	Pending		
<a href="#">5.6.4</a>	Integration	Integration testing content	Pending		
<a href="#">5.6.5</a>	Integration	Verify integration test procedure	Pending		
<a href="#">5.6.6</a>	Integration	Conduct regression tests	Pending		
<a href="#">5.6.7</a>	Integration	Integration test record contents	Pending		We can help write python scripts to automate the generation of these test records from the CI runs (we have some scripts we've accumulated from past projects that we can use as a starting place).
<a href="#">5.6.8</a>	Integration	Record bugs that are found during software-integration testing	Pending		
<a href="#">5.7.1</a>	System	Establish tests for software requirements	Pending		
<a href="#">5.7.2</a>	System	Record bugs that are found during system testing	Pending	This should be handled by existing QMS processes.	
<a href="#">5.7.3</a>	System	Retest after changes	Pending		
<a href="#">5.7.4</a>	System	Verify software system testing	Pending	Need to ensure each software requirement can be traced back to its verification.	We can help implement the tooling needed to enforce full traceability using a CI server.
<a href="#">5.7.5</a>	System	Test record contents	Pending	This should be handled by existing QMS processes.	
<a href="#">5.8.1</a>	Release	Ensure software verification is complete	Pending		This can be checked in the software release checklist.
<a href="#">5.8.2</a>	Release	Document known residual anomalies	Pending		See suggestions
<a href="#">5.8.3</a>	Release	Evaluate known residual anomalies	Pending		This can be checked in the software release checklist.

Section	Category	Description	Status	Status justification	How to fill the gap
<a href="#">5.8.4</a>	Release	Document version	Pending		Doing this should be included in the release checklist.
<a href="#">5.8.5</a>	Release	Document build process	Pending		
<a href="#">5.8.6</a>	Release	Ensure all activities complete	Pending		This can be checked in the software release checklist.
<a href="#">5.8.7</a>	Release	Archive software	Pending		
<a href="#">5.8.8</a>	Release	Assure repeatability of software releases	Pending		
<a href="#">6.1.a</a>	Maintenance	Procedure to gather customer feedback	Pending	It's assumed this will be handled above the software team	
<a href="#">6.1.b-f</a>	Maintenance	Procedure to determine whether feedback is a problem, documenting it, fixing it, etc.	Pending		See note at 5.1.1
<a href="#">6.2.1.1</a>	Maintenance	Monitor feedback	Pending		
<a href="#">6.2.1.2</a>	Maintenance	Document and evaluate feedback	Pending		
<a href="#">6.2.1.3</a>	Maintenance	Evaluate problem report's affects on Safety	Pending	This should likely be handled by software engineers participating in the system-level risk assessment procedures.	
<a href="#">6.2.2</a>	Maintenance	Use problem resolution process	Pending		
<a href="#">6.2.3</a>	Maintenance	Analyze change requests	Pending	This is being done informally, but the procedure will need to be documented.	See suggestions
<a href="#">6.2.4</a>	Maintenance	Change request approval	Pending	This is being done informally, but the procedure will need to be documented.	See suggestions in 6.2.3
<a href="#">6.2.5</a>	Maintenance	Communicate to users and regulators	Pending		
<a href="#">6.3.1</a>	Maintenance	Implement modifications	Pending	Software maintenance activities will be the same as the software development activities.	

 Section	 Category	 Description	 Status	 Status justification	 How to fill the gap
<a href="#">6.3.2</a>	Maintenances	Re-release modified software system	Pending	Software maintenance activities will be the same as the software development activities. In particular, we will follow the same release checklist as is used in software development.	
<a href="#">7.1.1</a>	Risk	Identify software items that could contribute hazardous situations	Pending		See suggestions in 5.1.7
<a href="#">7.1.2</a>	Risk	Trace sequences of events leading from the software item to the hazardous situations	Pending		See suggestions in 5.1.7. As with other traceability requirements, it could make sense to check this in the CI runs. We could help set up the tooling for this.
<a href="#">7.1.3</a>	Risk	Evaluate published SOUP anomaly list	Pending		See suggestions in 5.1.7 and in this item. We can help write this process.
<a href="#">7.1.4</a>	Risk	Document potential causes	Pending		See suggestions in 5.1.7
<a href="#">7.1.5</a>	Risk	Document sequences of events	Pending		See suggestions in 5.1.7
<a href="#">7.2.1</a>	Risk	Define risk control measures	Pending		See suggestions in 5.1.7
<a href="#">7.2.2</a>	Risk	Risk control measures in software	Pending		See suggestions in 5.1.7
<a href="#">7.3.1</a>	Risk	Verify risk control measures	Pending		Since risk control measures are treated as new features, verification shall be handled the same way all other software is verified per section 5.
<a href="#">7.3.2</a>	Risk	Document new sequences of events	Pending		See suggestions in 5.1.7. This will fulfilled with using the same activities that fulfill 7.1.1, 7.1.2, 7.1.4, and 7.1.5.
<a href="#">7.3.3</a>	Risk	Document traceability	Pending		See suggestions in 5.1.7. We can also provide Python scripts that will check traceability throughout development as part of the CI server.

Aa Section	Category	Description	Status	Status justification	How to fill the gap
<a href="#">7.4.1</a>	Risk	Analyze changes to medical device software with respect to safety	Pending		See suggestions in 5.1.7.
<a href="#">7.4.2</a>	Risk	Analyze impact of changes on existing risk control measures	Pending		See suggestions in 5.1.7. This will be fulfilled with using the same activities that fulfill 7.1.1, 7.1.2, 7.1.4, and 7.1.5.
<a href="#">7.4.3</a>	Risk	Perform risk management activities	Pending		See suggestions in 5.1.7. This will be fulfilled with using the same activities that fulfill 7.1.1, 7.1.2, 7.1.4, and 7.1.5.
<a href="#">8.1.1</a>	Configuration Management	Establish means to identify configuration items	Pending		
<a href="#">8.1.2</a>	Configuration Management	Identify SOUP	Pending		
<a href="#">8.1.3</a>	Configuration Management	Identify system configuration documentation	Pending		
<a href="#">8.2.1</a>	Configuration Management	Approve change requests	Pending		We suggest tying approvals to sprint planning meetings. I.e., by adding issues to the current sprint they are approved.
<a href="#">8.2.2</a>	Configuration Management	Implement changes following software development activities	Pending		We suggest meeting this through training the software engineers to follow a certain checklist during development. Pull reviewers can verify the checklist is being followed.
<a href="#">8.2.3</a>	Configuration Management	Verify changes	Pending		We suggest following a formal pull request review checklist.
<a href="#">8.2.4</a>	Configuration Management	Provide means for traceability of changes	Pending		We suggest recording change requests and problem reports as GitHub issues. This requirement will be met so long as (1) each pull request is tied to a GitHub issue, and (2) each pull request is reviewed this requirement is met.
<a href="#">8.3</a>	Configuration Management	Configuration status accounting	Pending		



Aa Section	☰ Category	☰ Description	⬇ Status	☰ Status justification	☰ How to fill the gap
<a href="#">9.1</a>	Problems	Prepare problem reports	Pending		I can draft a procedures to comply with this section. I'd work with QA/RQ to create a procedure that meshes well with the relevant system-level procedures. A simple approach is to track software problem reports using GitHub Issues. A GitHub Issue template can standardize the sections that need to go into the reports.
<a href="#">9.2</a>	Problems	Investigate the problem	Pending		I can draft a procedures to comply with this section. I'd work with QA/RQ to create a procedure that meshes well with the relevant system-level procedures. I propose that software engineers be allowed to evaluate and document whether an issue is "low risk" on their own OR to escalate the issue to the full risk management team.
<a href="#">9.3</a>	Problems	Advice relevant parties	Pending		Software engineers would need to inform the appropriate regulatory person who can decide what regulatory authorities or customers need to be notified of the problem.
<a href="#">9.4</a>	Problems	Use change control process	Pending		Change requests that are written due to problem reports will be handled like all other change requests, per section 5.
<a href="#">9.5</a>	Problems	Maintain records	Pending		If problem reports are stored as GitHub Issues, this will be handled automatically by following the sprint planning activities.
<a href="#">9.6</a>	Problems	Analyze problems for trends	Pending		We suggest that this analysis be performed as part of the release process or periodically as part of post-market surveillance.
<a href="#">9.7</a>	Problems	Verify software problem resolution	Pending		This will be handled automatically by following the existing software development activities.
<a href="#">9.8</a>	Problems	Test documentation contents	Pending		This will be handled automatically by following the existing software development activities.