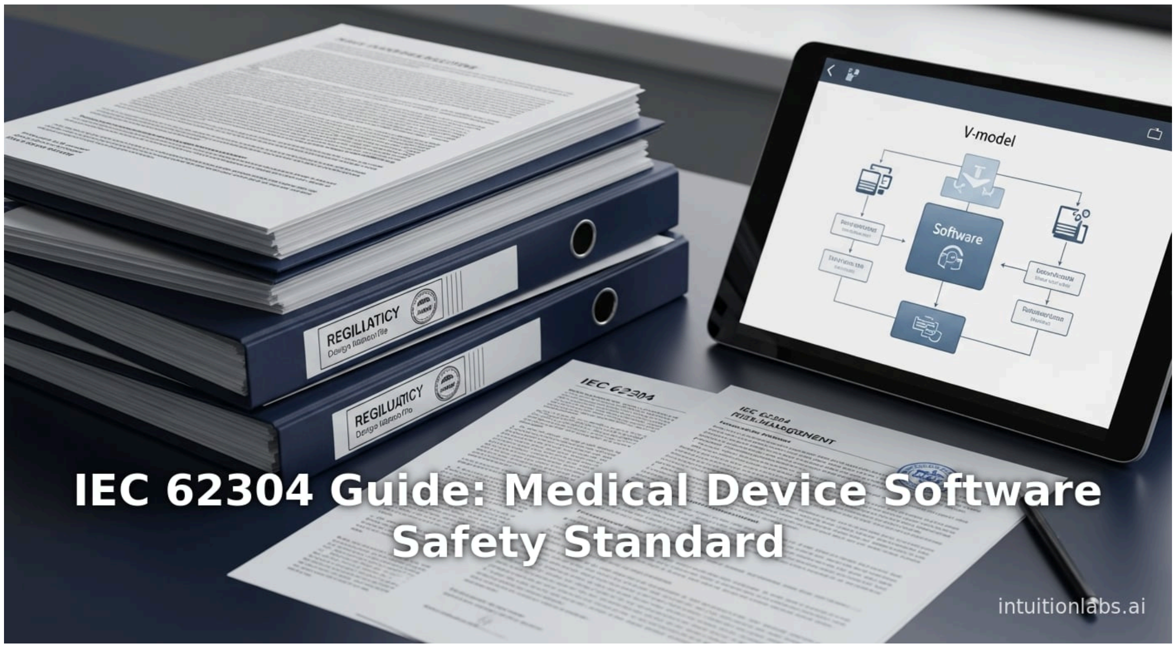
IEC 62304 Guide: Medical Device Software Safety Standard

By Adrien Laurent, CEO at IntuitionLabs • 2/1/2026 • 65 min read

iec 62304 medical device software software lifecycle processes samd iso 14971 fda software guidance soup software medical device compliance



Executive Summary

Medical device software (classified as either Software as a Medical Device [SaMD] or *Software in a Medical Device*[SiMD]) plays an increasingly critical role in modern healthcare, driving innovation and improving patient outcomes. However, high-profile failures (e.g., the Therac-25 radiation overdose incidents causing multiple patient deaths ([1] smartbear.com), and a 2000 treatment-planning software error in Panama that killed 21 of 28 patients ([2] www.mddionline.com)) have underscored the dire consequences of unsafe software development practices. In response, regulators and industry globally have adopted **IEC 62304** – *Medical device software – Software life cycle processes* – as the authoritative standard for software development and maintenance. Widely recognized by agencies such as the U.S. FDA, EU authorities, and the UK MHRA, IEC 62304 defines a risk-driven software life cycle framework that scales rigor (documentation, testing, validation) to the potential harm of software failure ([3] www.qualityfwd.com) ([4] 8foldgovernance.com). Compliance with IEC 62304 is typically expected for CE marking in Europe and as part of FDA submissions (it is listed as a *recognized consensus standard* by FDA ([5] www.accessdata.fda.gov)).

This comprehensive report examines IEC 62304 in depth, including its **scope, safety classification scheme, and core processes**. It situates the standard within the broader regulatory environment (e.g. EU MDR, FDA regulations, ISO 13485 QMS, ISO 14971 risk management) and compares it to related standards. We analyze real-world cases to illustrate the stakes of software safety, present expert perspectives on best practices, and discuss implementation challenges (including managing legacy software or “SOUP”, *Software of Unknown Provenance* ([6] www.softwarecpr.com), and adopting agile methods within a regulated framework ([7] www.riskmanager.net) ([8] www.riskmanager.net)). The report also highlights emerging issues: for example, a draft Second Edition of IEC 62304 is broadening scope to **all health software** (not just regulated medical devices) and incorporating guidance for AI/ML-based software ([9] 8foldgovernance.com) ([10] www.nsf.org). These expansions align IEC 62304 with trends in software-enabled healthcare and give developers a unified framework even as technology (e.g. AI, internet-connected devices, cybersecurity threats) evolves.

Key findings include:

- **Fundamental Role of IEC 62304:** It is the *de facto* global standard for medical device software life cycle processes, emphasizing traceable requirements, risk-managed design, systematic verification/validation, and controlled maintenance. It explicitly links to quality management (ISO 13485) and risk management (ISO 14971) ([11] quickbirdmedical.com) ([12] www.qualityfwd.com). Regulatory bodies expect developers to follow its processes and will scrutinize documentation and justifications “to prove the software was developed in a state of control” ([13] www.qualityfwd.com).
- **Risk-Based Classification:** Software is classified as Class A, B, or C based on potential harm from failure (www.iss-ag.ch). Class A (lowest risk) means failures cause no injury or only minimal, acceptable risk; Class B indicates non-serious injury; Class C indicates the potential for serious injury or death (www.iss-ag.ch). The depth of development controls (e.g. review detail, testing extent) scales with class – Class C requires the most rigorous processes. (Table 1 summarizes safety classes, examples, and analogous FDA “Level of Concern”.)
- **Comprehensive Lifecycle Processes:** IEC 62304 mandates a full software life cycle: detailed planning, requirements definition, architecture/design, coding, integration, testing, release, plus separate maintenance, risk management, configuration management, and problem-resolution processes ([14] quickbirdmedical.com) ([15] visuresolutions.com). These processes are designed to ensure software safety and to create a complete audit trail (e.g., traceability of requirements to design to tests) for regulatory review ([16] www.riskmanager.net) ([13] www.qualityfwd.com).
- **Regulatory Alignment:** In Europe, IEC 62304 provides “presumption of conformity” when harmonized standards are used under the MDR/IVDR. The FDA formally recognizes IEC 62304:2006+Amd1:2015 for submissions ([5] www.accessdata.fda.gov). Thus complying with IEC 62304 facilities regulatory approval without reinventing processes. It also dovetails with other standards (e.g., IEC 60601-1-4 for programmable systems safety, IEC 62366 for usability) to address broader safety aspects of software medical devices ([11] quickbirdmedical.com) ([17] bluegoatcyber.com).

- **Implementation Considerations:** Effective compliance requires strong quality management (ISO 13485) and integration of risk analysis (ISO 14971) ([11] quickbirdmedical.com) ([12] www.qualityfwd.com). Experts note that IEC 62304 “cannot be done with informal or ad hoc processes” ([18] www.qualityfwd.com); rather, organizations must establish or extend their QMS to cover software specifics (SOPs, templates, eQMS tools) ([19] www.qualityfwd.com) ([20] www.qualityfwd.com). Agile development is compatible but requires documented iteration and traceability built into sprints ([7] www.riskmanager.net) ([16] www.riskmanager.net).Integrating third-party code (SOUP) demands including its failure modes in hazard analyses ([6] www.softwarecpr.com) ([21] www.softwarecpr.com).
- **Case Studies:** The report examines multiple real-world examples: from the Therac-25 and other historical software failures ([1] smartbear.com) ([2] www.mddionline.com), to modern recalls (e.g., Abbott’s Libre3 glucose sensors linked to injuries ([22] apnews.com)). These cases illustrate how software bugs or inadequate validation can have life-or-death consequences, reinforcing the necessity of IEC 62304-type processes.
- **Emerging Trends:** The pending IEC 62304 second edition is expanding scope to *all* health-related software, replacing the old A/B/C classes with two “rigor levels”, and adding specific guidance for AI/ML software ([9] 8foldgovernance.com) ([23] www.nsf.org). Aligning with FDA/IMDRF guidance, it also removes normative references to ISO 13485/14971 (since non-device health apps may not be covered by those) while clarifying that compliant organizations should continue following them ([24] 8foldgovernance.com) ([25] 8foldgovernance.com). Cybersecurity is increasingly treated as integral (e.g. FDA’s guidance requires security by design), and IEC 62304’s process framework provides a foundation to build in security controls ([26] bluegoatcyber.com) ([27] bluegoatcyber.com). Future directions include handling AI model lifecycle, post-market monitoring of software changes, and harmonization with new regulations (e.g. EU AI Act, future FDA digital health frameworks).

In sum, **IEC 62304 is the cornerstone** for medical device software safety. Its rigorous, evidence-based approach addresses known failure modes and guides manufacturers to systematically manage risks. Ensuring compliance with IEC 62304 (especially its upcoming revision) is critical for patient safety and regulatory success. As software-driven healthcare continues to grow, understanding and applying the principles of IEC 62304 will remain essential.

Introduction and Background

The Digital Transformation of Medical Devices

In recent decades, medical devices have undergone a profound transformation as software becomes embedded in nearly every modality of healthcare. Modern devices – from implantable pacemakers and infusion pumps to diagnostic imaging systems and smartphone apps – rely on sophisticated software to function. This trend is accelerating: analysts predict the Software as a Medical Device (SaMD) market in Europe will grow from about €440 million in 2024 to over €1.4 billion by 2033 ([28] punktum.net). Globally, software-driven devices promise improved diagnostics, personalized therapies, and remote monitoring capabilities. However, they also introduce new hazards. Unlike mechanical failures, software defects can produce unpredictable and potentially catastrophic outcomes if not managed methodically.

Tragic accidents in the 1980s and beyond illustrated the stakes. The Therac-25 linear accelerator (1985–1987) delivered massive radiation overdoses due to software race-condition bugs, killing at least three patients and injuring others ([1] smartbear.com) ([29] smartbear.com). In 2000, an unsafe update to radiotherapy planning software in Panama led to 28 cancer patients receiving fatal overdoses of gamma radiation (21 ultimately died) ([2] www.mddionline.com). More recently, a medication infusion pump software error in 2015 (CareFusion Alaris pump) delayed drug delivery, prompting a Class I recall ([30] www.mddionline.com); ventilator software flaws were similarly recalled that year. These and other incidents (Table 10) drove home the fact that software errors can be *every bit as deadly* as hardware malfunctions ([31] www.mddionline.com).

Regulators responded by updating frameworks. Early device directives and standards (e.g. IEC 60601 electrical safety) did not specifically address software life cycle. In the late 1990s, voluntary guidance like AAMI’s Software Validation series and FDA white papers emerged. Ultimately, in 2006 the International Electrotechnical Commission (IEC) published **IEC 62304**, an international standard dedicated to *medical device software life cycle processes*. IEC 62304 provided the first comprehensive, consensus-based requirements for developing and maintaining safe medical software. It built on fundamental quality principles (ISO 13485 for QMS, ISO 14971 for risk) while filling the software-specific gap.

IEC 62304:2006 was later amended in 2015 (released as IEC 62304:2006+Amd1:2015, formally Edition 1.1) ([32] visuresolutions.com). The amendment introduced key enhancements: for example, it mandated a risk-based approach to software safety classification and clarified how to handle legacy software and third-party code (SOUP) ([32] visuresolutions.com) ([6] www.softwarecpr.com). Both editions of IEC 62304 now carry legal weight via regulatory recognition: the FDA *recognizes* IEC 62304:2015 as a consensus standard for submissions ([5] www.accessdata.fda.gov), and the EU Medical Device Regulation (MDR 2017/745) and In Vitro Diagnostic Regulation (IVDR 2017/746) allow using IEC 62304 to achieve a *presumption of conformity* with software requirements in Annex I. In practice, auditors worldwide expect compliance with IEC 62304 as evidence of a controlled software process ([4] 8foldgovernance.com) ([3] www.qualityfwd.com).

This report dives deep into IEC 62304: examining its history and rationale, explicating its contents (scope, processes, classification, and supporting standards), and discussing its implementation. We also analyze case studies, drawing on data (e.g. recall statistics) and expert commentary to highlight best practices and pitfalls. We conclude with emerging trends, including the draft second edition and the impact of new technologies (AI, cybersecurity, connected health). Through this comprehensive review, readers will gain a thorough understanding of *why* IEC 62304 matters, *how* to apply it, and *where* medical software regulation is headed.

Scope and Purpose of IEC 62304

IEC 62304 establishes a **framework for the safe development and maintenance of medical device software**. Its official title is *"Medical device software – Software life cycle processes"*. The standard covers any software that is either a medical device in itself or part of a medical device ([33] quickbirdmedical.com). In broad terms, it applies to:

- **Standalone medical software (SaMD):** Software running on general-purpose hardware, mobile devices, or in the cloud, whose intended purpose is medically therapeutic, diagnostic, or health-related ([34] quickbirdmedical.com) ([35] www.fda.gov). Examples include clinical decision-support apps or chronic monitoring software.
- **Embedded/device software (SiMD):** Software integrated into a physical medical product (e.g. CT scanners, infusion pumps, pacemakers) ([33] quickbirdmedical.com).
- **Firmware systems:** Software within a device's embedded control systems, often with real-time constraints.
- **Machine-learning modules** (under development): As discussed below, recent updates include guidance for AI/ML components.

IEC 62304's *Application Area (Clause 1)* explicitly states it covers the entire software life cycle – from planning through retirement – for any medical device software, whether new or legacy ([36] quickbirdmedical.com). Even software that is not yet regulated (e.g. in development) or software used in GMP processes is subject to the same principles. The standard does **not** prescribe a particular development model (waterfall, agile, etc.), but it does require that whatever process is used be well-documented, repeatable, and controlled within a quality system.

A key feature is the **risk-based approach**. IEC 62304's requirements scale with the severity of potential harm from software failure. As one industry source notes, *"IEC 62304 matters because it is a harmonized standard…Compliance is risk-based, meaning the effort and documentation required is scaled according to the potential harm the software could cause if it fails"* ([3] www.qualityfwd.com). This means that low-risk Class A software needs less documentation and fewer formal reviews than high-risk Class C software (discussed further below).

In addition to lifecycle activities, IEC 62304 explicitly references related standards. For instance, Clause 4 (*General Requirements*) mandates that the manufacturer have a quality management system (e.g. ISO 13485) and perform risk management per ISO 14971 ([11] quickbirdmedical.com). Consequently, IEC 62304 is designed to be used *with* these complementary standards: ISO 13485 for overall system quality and ISO 14971 for managing hazards. (Notably, the draft second edition will broaden scope beyond medical devices, so it removes normative references to ISO 13485/14971, although following them remains best practice ([24] 8foldgovernance.com).)

By enforcing a rigorous, process-oriented approach, IEC 62304 helps manufacturers avoid common software development pitfalls (such as inadequate testing, undocumented requirements, and uncontrolled changes) ([37] quickbirdmedical.com). It also provides auditors with objective criteria: a completed IEC 62304-compliant life-cycle yields artifacts like software requirements specs, design reviews, and test reports, which collectively form a traceable design history. As one quality consultant observes, *“a compliant QMS is the key to proving to auditors and regulators that your software was developed in a state of control from its initial concept all the way through to its final retirement”* ([13] www.qualityfwd.com).

The importance of IEC 62304 in regulation can be seen in its endorsement by authorities: The FDA lists IEC 62304:2006+Amd1:2015 as a recognized consensus standard for medical device software development ([5] www.accessdata.fda.gov), and regulators in the UK and EU consider it “well-recognized as the best approach” to software safety ([4] 8foldgovernance.com). In practice, even if compliance with IEC 62304 is not legally *mandatory* (CE marking does not *require* written conformance to specific standards), using it greatly simplifies demonstrating conformity with regulatory requirements, as conformity is assumed when harmonized standards are applied.

Safety Classification (IEC 62304:2006/2015)

A core concept in IEC 62304 is the **Software Safety Class**, which categorizes software by the potential severity of harm from its malfunction. There are three classes:

- **Class A:** The software cannot contribute to a situation that results in injury (or it does, but those injuries are considered acceptable and not even non-serious). In essence, Class A is the *lowest risk* category. These are features whose failure might cause inconvenience but not harm. An example might be a non-essential display feature or logging function. Even for Class A, the manufacturer must apply the standard’s core processes, but less rigor is required in documentation and verification.
- **Class B:** Software where a failure could contribute to a *non-serious injury* or non-life-threatening harm. Think functions like basic alarms or record-keeping where an error might inconvenience the user or require minor medical attention but is unlikely to be life-threatening. Moderate rigor is mandated: more testing and documentation than Class A, but not the maximum.
- **Class C:** Software whose failure could lead directly to *serious injury or death*. These are the highest-risk functions – for example, control of a life-sustaining therapy (ventilator breath control, insulin dose calculation) or safety features (emergency shut-off). Class C demands the most stringent controls: comprehensive reviews, exhaustive testing, and extensive documentation.

IEC 62304’s rules for determining the class are tied to risk analysis. In practice, one conducts a hazard analysis (often according to ISO 14971 methodology), then applies IEC 62304 criteria:

> **“When software failures cannot result in patient/user exposure to a hazard, or result only in acceptable residual risks, the software is Class A. When residual risks from a software failure become unacceptable, the software is Class B or C depending on whether the resulting injury is considered non-serious or serious, respectively.”** (www.iss-ag.ch).

In other words (using the standard’s own logic):

- If the failure contributes no hazardous condition or only an adequately controlled one ⇒ **Class A**.
- If the failure could contribute to an uncontrolled hazard leading to *non-serious* injury ⇒ **Class B**.
- If it could contribute to *serious* injury or death ⇒ **Class C**.

(IEC 62304's definitions of "serious" injury align with ISO 14971, and "harm" is interpreted in the broad sense used in risk management ([38] www.nsf.org).) Table 1 summarizes these classes with illustrative examples and compares them to FDA's level-of-concern terminology.

IEC 62304 Class	Potential Harm from Software Failure	Example	FDA "Level of Concern"
A	No injury or only *acceptable* (minor) risk. Software failures *cannot* cause a hazard that leads to harm.	E.g. software modules for non-therapeutic monitoring, simple user interface functions.	Minor (Level 1) – worst-case impact is minor inconvenience or inconvenience-only.
B	Software failure *could* contribute to a hazardous situation resulting in **non-serious injury***.	E.g. an alarm function or intermediate data processing where error might cause transient or treatable harm.	Moderate (Level 2) – impact is non-life-threatening but requires medical care.
C	Software failure *could* contribute to a hazardous situation resulting in **serious injury or death***.	E.g. control software for ventilator, critical drug dosage calculation.	Major (Level 3) – impact is life-threatening or fatal if not managed.

"Non-serious" vs. "serious" per IEC is aligned with ISO 14971's definitions, generally meaning absence or presence of life-threatening consequences.

This risk-based classification drives the development process. For higher classes, IEC 62304 imposes increased documentation, design reviews, and verification activities. The standard's General Requirements section (Clause 4) explicitly ties the software's class to the rigor of subsequent processes. For instance, Class C software must produce a detailed Software Requirements Specification (SRS) and verified design for every software item, whereas for Class A less detailed artifacts may suffice ([11] quickbirdmedical.com).

IEC 62304 also requires one to assume that *"software failure occurrence shall be assumed to be certain (probability = 100%)"* for the purpose of classification (www.iss-ag.ch). This means when analyzing risks, one treats any software bug as a definite event, focusing instead on how controls mitigate its impact. In short, the class determination is conservative: if any plausible software bug could lead to unacceptable harm, the higher class is assigned. This policy ensures rigorous treatment of uncertainty (justified given the complexity inherent in software) (www.iss-ag.ch).

It is worth noting that regulatory agencies sometimes use parallel classification schemes. For example, the FDA has historically categorized device software by *"Level of Concern"* (LoC) – minor, moderate, or major – which roughly correspond to IEC's Class A/B/C ([39] visuresolutions.com) ([26] bluegoatcyber.com). Under recent FDA guidance for SaMD, LoC is determined by the role of the software in driving healthcare decisions and potential patient harm. While different terminologies are used, the underlying concept is the same: higher risk demands more assurance.

Assigning the correct class is crucial. A wrong assignment (e.g. designating truly critical software as Class B instead of C) could leave insufficient controls in place, creating safety vulnerabilities (www.iss-ag.ch). Therefore, manufacturers often over-classify ("erring on the side of safety") to ensure compliance. Once the class is set, the title of IEC 62304 is effectively *"Software Class X life cycle processes"*, where X is A, B, or C. The rest of the standard's requirements are interpreted through the lens of that class.

Key Processes of IEC 62304

IEC 62304 prescribes **five core process areas** for software life cycle and maintenance, each subdivided into specific activities. Table 2 outlines these processes with their fundamental objectives and key activities. The standard is structured so that Clauses 5–9 align with these processes:

- **Software Development Process (Clause 5):** Encompasses initial development from concept to release. This includes Software Development Planning (5.1), Requirements Analysis (5.2), Architecture Design (5.3), Detailed Design (5.4), Unit Implementation and Verification (5.5), Integration and Integration Testing (5.6), System Testing (5.7), and Release (5.8). Together, these form a waterfall-like lifecycle (which can be executed iteratively). At each stage, specific documentation and verification tasks are required. For example, one must verify that implemented software units correctly fulfill their design (Unit Testing) and that assembled software meets the requirements (Integration/System Testing) ([40] quickbirdmedical.com). Software Release (5.8) mandates versioning and release notes.

- **Software Maintenance Process (Clause 6):** A selective, streamlined process for post-market changes. When a device is in use, any change (bug fix, minor enhancement) must go through maintenance planning, re-analysis of risk, implementation, and testing. Because urgency may preclude full redeployment planning, Clause 6 allows a reduced process (e.g. limited verification) for minor fixes, but still demands risk analysis and traceability. In effect, maintenance is a capped version of development principals to expedite urgent corrections.
- **Software Risk Management Process (Clause 7):** This process integrates ISO 14971 risk management into software. It requires identifying potential hazards due to software failure, estimating their risk by combining hazard severity with probability, and specifying software safety requirements or controls to mitigate those risks. Notably, IEC 62304 calls for incorporating all risk-derived software requirements into the SRS ([41] www.riskmanager.net). The standard also mandates that if external controls (e.g. hardware interlocks) exist, those are considered in the software's risk analysis, but only controls *outside* the failing software are allowed to reduce the assessed risk (www.iss-ag.ch).
- **Software Configuration Management (Clause 8):** Ensures integrity of the evolving software artifacts. It requires version control of source code, documentation, and builds. Every file (SRS, design diagrams, test scripts, source code, executables) must be under configuration control, with change procedures. This process ensures that any release can be traced back to a specific set of code and documents, which is vital for investigation of issues (e.g., in a recall, one can identify exactly which build was in the field).
- **Software Problem Resolution (Clause 9):** A formal defect and anomaly handling process. It requires logging any software aspect that does not meet requirements (from code bugs to documentation errors) as a "software problem report". Each item must be evaluated for severity, root cause, and appropriate action (repair, workaround). For Class B/C systems, all anomalies must be analyzed and resolved; Class A allows possibly fewer formalities. The process also requires verifying and closing problems, and feeding lessons learned back into coding or risk analysis.

Each of these processes generates work products: plans, requirement documents, design records, test protocols, issue logs, etc. In aggregate, this documentation forms the Software *Design History File* (DHF), which regulators will scrutinize. A typical IEC 62304 development yields a tight trace: one can trace each software requirement (derived from risk analysis) through design, implementation, and testing. For example, risk-derived safety functions are codified as requirements in the SRS, implemented in code, and accompanied by dedicated verification tests. Attesting to authorities "we did that safety analysis according to 14971" without showing how those requirements entered the software is insufficient; hence IEC 62304 demands explicit traceability ([16] www.riskmanager.net).

The standard does not *dictate* exactly *how* to implement processes. For example, Annex B of IEC 62304 (non-normative) explicitly states that agile development is compatible: plans may be iterative, and processes may recur as development progresses ([7] www.riskmanager.net). One risk manager notes: *"planning is an iterative ACTIVITY that should be repeatedly reviewed and updated as development progresses"* ([7] www.riskmanager.net). In practice, teams can use Scrum or other agile methods as long as they produce the required deliverables (e.g. backlog tied to hazards, sprint review of requirements). What matters is that nothing is skipped: every requirement must be verified, and every change must be controlled and justified.

Table 2 summarizes these processes, emphasizing the flow from planning to problem resolution. Each process may be implemented via any lifecycle model (V-model, agile iterations, etc.), but must be *defined* in the QMS. The table entries highlight the intention and example work products at each stage, with references to the standard's clauses and guidance sources.

Table 2: Overview of IEC 62304 Processes and Activities

Process (Clause)	Goal / Main Activities	Key Deliverables & Notes
5 – Software Development	Plan, design, implement and test new software. Activities include software development planning (scope, resources, interfaces) ([7] www.riskmanager.net); requirements analysis (detail functional & safety requirements) ([14] quickbirdmedical.com); software architecture and detailed design; coding (unit implementation); unit verification tests; integration of units; system-level testing; and release preparation.	Software Development Plan; **Software Requirements Spec (SRS)**; Architecture and Design docs; Unit test reports; Integration/System test reports; Release notes. IEC 62304 mandates verification at each phase (unit tests, integration tests, etc.) ([40] quickbirdmedical.com) ([16] www.riskmanager.net).
6 – Software Maintenance	Manage post-market changes (bug fixes, security patches, minor enhancements). Perform impact/risk analysis of requested changes, plan maintenance, implement, verify, and document solutions. A scaled-down but controlled process to allow rapid fixes.	Updated versions of design docs and test results for the changes; maintenance plan; change control records. Must still re-evaluate risk and update SRS or risk analysis when needed.

Process (Clause)	Goal / Main Activities	Key Deliverables & Notes
7 – Software Risk Management	Identify and control risks due to software. Perform hazard analysis assuming software failure (probability=100%) and only external risk controls (www.iss-ag.ch). Derive software safety requirements to mitigate unacceptable risks, and integrate those into SRS. Verify that safety features/control measures meet intended function.	Risk Management File (hazard analysis, risk evaluation, risk control rationales) (www.iss-ag.ch); updated SRS with risk-derived requirements. Evidence that each software hazard's risk is acceptably controlled per ISO 14971.
8 – Configuration Management	Ensure integrity and traceability of software artifacts. Establish version control for source code, documentation, test scripts, executables, etc. Manage branching, baselines, and change control.	Configuration Management Plan; Version-controlled repository of all SW artifacts; change control records. (Compare every release to a baseline; document build environment.)
9 – Problem Resolution	Handle software anomalies and defects. Log every discovered bug or failure (in development or field). Assess severity, repeat/incidence, assign corrective action (fix, monitor, defer). Verify and close out each problem. Update other processes if needed (e.g. risk analysis).	Software Problem Reports (bug reports) with analysis of cause and impact; Verification of fixes; Updated hazard/risk analysis if a new risk was identified. Ensures closure of each issue to prevent recurrence.

Table 2 Note: These processes apply to *all* classes (A–C), but IEC 62304 allows simpler documentation for Class A in certain areas. For example, Class A might not require formal SRS annexes or design reviews to the same extent as Class C ([3] www.qualityfwd.com). However, even Class A still demands a documented plan, risk consideration, unit tests, and traceability.

General Requirements (Clause 4)

Before embarking on the specific processes, IEC 62304's Clause 4 imposes foundational QMS and risk management requirements. It requires **three general prerequisites**:

1. **Quality Management System (QMS)**: The manufacturer must have an established QMS for medical devices. While IEC 62304 does not detail the QMS itself, it notes that compliance with ISO 13485 is a way to meet this. In effect, IEC 62304 assumes that design controls (as in ISO 13485:2016 Clause 7.3 or FDA 21 CFR 820.30) are in place. The QMS should ensure that software processes are planned, executed, and documented consistently ([11] quickbirdmedical.com) ([12] www.qualityfwd.com).

2. **Risk Management (ISO 14971)**: A top-level risk management process per ISO 14971 must be in place. IEC 62304 does not itself derive risk, but it incorporates ISO 14971 by reference. The manufacturer must manage software hazards *through that system*. In practice, this means that hazards and risks from software are identified, evaluated, and controlled just like hardware risks ([11] quickbirdmedical.com) ([15] visuresolutions.com). Indeed, later clauses rely on ISO 14971 – for example, the risk controls and "severity of harm" come from that standard's definitions ([38] www.nsf.org).

3. **Software Safety Classification (Classes A/B/C)**: As discussed above, each software item must be assigned a safety class (4.3). The class determines the rigor of documentation required (higher class ⇒ more documents, reviews, tests). The manufacturer has freedom in partitioning software into "items" and classes. For example, a single app might be split into one Class A subsystem (e.g. a diagnostic plugin) and one Class C subsystem (the main therapy control) to isolate risks ([42] quickbirdmedical.com). IEC 62304 provides a decision tree for classification (Figure 1 in the standard); essentially, any uncertain case of unacceptable risk is escalated to the higher class.

By enforcing these prerequisites, IEC 62304 ensures that software is not developed in a vacuum. The broad quality framework (QMS) and risk processes are prerequisites, and software classification triggers the appropriate level of process control. In sum, Clause 4 declares *"the manufacturer must…"* do these things, setting the stage for the detailed processes that follow.

Integration with Quality and Risk Standards

IEC 62304 does not stand alone; it operates within an ecosystem of standards. This integration is crucial for effective compliance and best practices.

ISO 13485 – Quality Management

ISO 13485 (Medical devices – Quality management systems) is the generic standard for device manufacturers' QMS. Clause 4.1 of IEC 62304 explicitly references ISO 13485: it requires the manufacturer to "operate a quality management system" ([11] quickbirdmedical.com). In effect, all activities in IEC 62304 must be under a QMS umbrella. Notably, ISO 13485 Clause 7.3 (Design and development) overlaps substantially with IEC 62304's requirements: ISO 13485 demands design inputs/outputs, design review, verification, validation, and change controls, but doesn't specify the content for software. IEC 62304 can be viewed as *executing* ISO 13485's design controls specifically for software.

A QualityForward summary highlights this synergy: *"IEC 62304 focuses specifically on the software lifecycle, while ISO 13485 defines the requirements for the overall quality management system… The two standards are designed to work together in harmony"* ([12] www.qualityfwd.com). In practice, companies often integrate IEC 62304's processes into their ISO 13485 QMS. For example, the IEC 62304 Software Development Plan becomes a subset of the QMS's design plan. Traceability matrices link ISO 13485 action items to IEC 62304 work products, and change control procedures cover both hardware and software changes. Indeed, one QMS consultant notes that building an "IEC 62304 QMS" is essentially part of meeting ISO 13485's design documentation and control requirements ([19] www.qualityfwd.com).

Thus, compliance with ISO 13485 (often mandatory for CE marking) inherently supports IEC 62304: a strong QMS provides the mechanisms (document control, audits, CAPA, supplier controls, etc.) that software processes require. Conversely, IEC 62304 provides ISO 13485 with the specific guidance for *software* components that ISO lacks. For example, an ISO 13485 audit might check that a device's software design plan exists; an IEC 62304-focused audit will check that the plan includes software-specific risk analysis and verification plans.

ISO 14971 – Risk Management

ISO 14971 (Application of risk management to medical devices) is the international standard for device risk management. IEC 62304 integrates risk management in two ways:

- **Mandated linkage:** As noted, Clause 4.2 requires a risk management process per ISO 14971. This means all software hazards and associated harms must be assessed according to that standard's methodology ([11] quickbirdmedical.com). In effect, ISO 14971 defines risk (probability × severity) and decides what is "acceptable". IEC 62304 then says: *whatever ISO 14971's process finds, incorporate those results into software development.*
- **Embedded risk activities:** Clause 7 of IEC 62304 explicitly mirrors ISO 14971 phases but at the software level: identify software hazard causes (e.g. code defects), analyze severity (using ISO's harm categories), estimate probability (note IEC 62304 assumes 100% failure rate as worst-case, see sidebar), and evaluate risk. Crucially, IEC 62304 requires that *risk controls* for hazards involving software be implemented and verified as part of the software itself (unless infeasible). For example, a hazard analysis may reveal that missing data validation could cause misdiagnosis; the control might be additional input checks coded into the software – which then becomes a formal requirement in the SRS ([16] www.riskmanager.net).

IEC 62304 does **not** replace ISO 14971; rather it uses it as a companion. All hazard analyses should be performed under ISO 14971, and the results traced into IEC 62304 documentation. As one author emphasized, one of the "major confusions" is trying to separate IEC 62304 risk analysis from ISO 14971: in reality, "SOUP failure modes are in the same fault trees as your custom software so it would be easiest to analyze them simultaneously" ([21] www.softwarecpr.com).

Other Standards

Several other standards interact with IEC 62304 in important ways:

- **IEC 60601-1 (Medical electrical equipment):** Older editions of IEC 60601-1 did not cover software, but IEC 60601-1-4 (treated as part of IEC 60601-1) provides requirements for programmable electronic safety-related systems. It emphasizes separate requirements for software development and verification. IEC 62304 complements this by giving much more detailed SW process requirements. In practice, compliance with IEC 60601-1's 3.4 (safety systems) implies implementing something like IEC 62304 for the software portions.

- **IEC 62366 (Usability):** This standard covers usability engineering of medical devices. Its application predominantly concerns UI software. IEC 62366 requires identifying use-related risks and designing the UI accordingly. While IEC 62304 focuses on the software **engineering process**, IEC 62366 focuses on human factors. They intersect: for instance, use-related hazard mitigations (from IEC 62366 process) may become software requirements (e.g. confirmation dialogs, error messages) that IEC 62304-related processes must implement and verify.
- **IEC 82304-1 (Health Software):** Published in 2017, IEC 82304-1 addresses standalone health software quality and safety. It is largely aimed at software not intended as regulated medical devices (general wellness apps, diagnostic apps outside strict MD definitions). However, aspects of IEC 82304-1 overlap with IEC 62304 (risk management, testing, user interaction). The draft second edition of IEC 62304 explicitly aligns its scope with IEC 82304(-1) and other "health software" standards ([43] 8foldgovernance.com). Ultimately, IEC 62304 is intended for **regulated** devices, whereas IEC 82304-1 can apply to health software at large. Having consistent principles across both benefits developers who operate in dual spaces.
- **21 CFR 820 (FDA Quality System Regulation):** While US code is not a "standard," it imposes similar requirements on US device manufacturers (design controls, traceability, complaint handling). IEC 62304's documentation and processes fulfill many 820 requirements. For example, 21 CFR 820.30 requires design/input requirements and verification records – these are mirrored by IEC 62304's SRS and test reports. Thus, a company complying with IEC 62304 is effectively satisfying the relevant portions of 820 for software.

In summary, IEC 62304 sits at the nexus of software development, quality management, and risk management. It provides the *detailed execution* of design and verification controls for software that ISO 13485 and ISO 14971 define at a high level ([12] www.qualityfwd.com). Adopting IEC 62304 means integrating these standards: e.g. the IEC 62304 Software Development plan becomes part of the ISO 13485 Design and Development plan; every risk identified in IEC 62304's Section 7 should tie back to the ISO 14971 risk file. This integration ensures consistency and avoids gaps or duplications. As one industry commentary states, using IEC 62304 is *"the clearest way to demonstrate to regulators that you have followed a rigorous and repeatable process"* ([3] www.qualityfwd.com).

Regulatory Landscape and Global Perspectives

IEC 62304's impact and usage vary somewhat by region, but a common global theme is regulatory expectation of structured software processes. Below we outline how major regulators and regions treat medical device software and the role of IEC 62304.

United States (FDA)

In the U.S., the FDA regulates medical device software like any other device under the Federal Food, Drug, and Cosmetic Act. Software submitted as a device (e.g. SaMD) or as part of a device must follow applicable FDA directives, which for software requirements often refer in part to IEC 62304 or similar practices.

- **Recognized Consensus Standard:** The FDA maintains a list of recognized consensus standards for device submissions. IEC 62304:2006+Amd1:2015 is listed there for *"Medical device software – Software life cycle processes"* ([5] www.accessdata.fda.gov). When a manufacturer uses this standard, FDA reviewers may grant "special 510(k)" submissions or other regulatory relief. In effect, following IEC 62304 can reduce FDA documentation requests, since it signals adherence to best practices.
- **Software as a Medical Device (SaMD):** The FDA relies on IMDRF definitions for SaMD and has specific guidance on risk categorization and validation for standalone software. While IEC 62304 itself applies to both SaMD and SiMD, the FDA's SaMD guidance (continuously updated) emphasizes the need for verification, validation, and cybersecurity. Notably, the FDA's new *Computer Software Assurance (CSA)* initiative (2022–2023) is exploring more risk-based and agile-friendly approaches to software under 21 CFR requirements. Although CSA is still evolving, it does not eliminate the need for sound software life cycle processes – rather, it reframes how rigor is assigned. In the meantime, the conservative stance in the industry remains: treat FDA's expectations as at least as stringent as IEC 62304.

- **Post-Market and Recalls:** The FDA tracks software-related device warnings and recalls. In 2015 it reported software design errors as the most common root cause in device recalls (annual data) ([44] www.mddionline.com). Prominent recent cases include cybersecurity vulnerabilities (e.g. hospital networks hacked via medical devices) and software bugs in devices like blood-glucose monitors. For example, in late 2025 the FDA warned that Abbott's FreeStyle Libre 3 sensors were delivering incorrect low-glucose readings, causing seven reported deaths and over 700 injuries worldwide ([22] apnews.com). While hardware (sensor design) played a role, such incidents emphasize the need for robust software risk processes. Post-market surveillance (required under FDA regulations) ties back to IEC 62304 by mandating that any field issues enter the Problem Resolution process (Clause 9) and be analyzed for impact on software safety.
- **FDA Guidance:** The FDA has published multiple guidance documents that intersect with IEC 62304 principles. For software validation, the FDA's 2002 guidance (and subsequent updates) essentially parallel IEC 62304's intent. More recently, FDA's *General Principles of Software Validation* (2002) and *Cybersecurity Guidance* (2018 and 2022) emphasize traceability, documentation, and security – all aligned with IEC 62304's lifecycle focus ([26] bluegoatcyber.com). There is no single FDA regulation titled "IEC 62304," but the cumulative effect of FDA policy is that reviewers expect its core tenets to be met. Anecdotally, FDA auditors at inspections often ask to see the IEC 62304 software development plan, risk analysis for each software hazard, test results, and version control records.

In summary, compliance with IEC 62304 facilitates meeting FDA requirements under 21 CFR 820 and premarket review. Manufacturers developing in the U.S. should implement IEC 62304 processes or an equivalent compliant approach, and keep abreast of emerging FDA practices like CSA. The review environment is shifting toward validating software via an evidence-based, risk-proportional framework – IEC 62304 already embodies many such principles.

European Union (CE Marking, MDR)

The EU's regulatory framework for medical devices transitioned in May 2021 from the Medical Devices Directive (MDD) to the more stringent Medical Device Regulation (MDR, 2017/745) ([45] punktum.net). Software that qualifies as a medical device under MDR (including standalone software with a medical intended purpose) must comply with the regulation's essential requirements. IEC 62304 plays a key role in demonstrating that compliance:

- **MDR and SaMD:** Under MDR, any software intended for a medical purpose – whether incorporated in a device or standalone – is a "medical device software" (MDSW). The MDCG 2019-11 guidance (and its 2025 Rev.1 update) clarifies that software with a medical purpose must follow the full MDR conformity assessment, including likely Notified Body review (often Class IIa or higher) ([46] punktum.net). To obtain CE marking, manufacturers must show evidence of safety and performance per MDR Annex I and clinical evaluation (Annex X), including software considerations. Using harmonized standards is the usual way to cover MDR "General Safety and Performance Requirements". IEC 62304 is included on the EU's harmonized standards list (EN 62304:2006+A1:2015) for software life cycle processes, meaning that adherence to it provides a *presumption of conformity* with relevant regulation clauses on development and risk processes. In other words, if you develop your SW under IEC 62304, you can justify that the MDR's requirements for design and risk management have been met.
- **Classification (MDR Rule 11):** The MDR classifies all devices by risk class. A new rule (§6.3 of Annex VIII, often called *Rule 11*) specifically addresses software. Most standalone SaMD end up as Class IIa or higher in the EU, because software can drive clinical decisions or monitor vital parameters ([45] punktum.net). For example, diagnostic software or an app that influences treatment is often IIb or III. IEC 62304's internal A/B/C classes are separate from MDR classes. However, in practice, Class III MDR devices (highest regulatory risk) will almost always have critical Class C software components, while a Class I device might contain only Class A software. MDCG 2021-24 provides details on applying Rule 11. Importantly, the need for IEC 62304 does not strictly depend on MDR class: any device containing software subject to Directive 93/42/EEC or MDR is expected to follow IEC 62304's development controls (or an equivalent process) ([34] quickbirdmedical.com).
- **Notified Body Audits:** Notified Bodies under MDR will inspect manufacturers' documentation for alignment with IEC 62304. They will expect traceability from user needs/risk analysis through requirements, code, and tests ([16] www.riskmanager.net) ([13] www.qualityfwd.com). Common audit points include software planning (the plan should refer to standards like IEC 62304), the Risk Management File (integrated with software), evidence of version control (e.g. a source-control log), and records of software testing. Because MDR emphasizes post-market performance, software updates also require formal change control under IEC 62304's maintenance process, possibly with new clinical evaluation under the new MDR rules.

- **Harmonized Standards:** In the EU 2021–2026 period, the list of harmonized standards for the MDR is evolving. Commission decisions (e.g. 2021/1182, 2022/757, etc.) officially publish references to standards and amendments. EN 62304:2006+A1:2015 currently appears as harmonized for both MDR and IVDR. (As of early 2026 the EU has also begun listing IEC 62304's forthcoming second edition in draft.) In any case, the existence of a harmonized (EN) version means manufacturers can cite *EN 62304* in CE Technical Documentation. It also means those standards have legal weight: failure to apply a listed standard leaves manufacturers needing strong justification for any divergence.

In short, IEC 62304 is effectively *required practice* for CE marking of software: while legally the MDR does not mandate *which* standards to use, compliance with harmonized standards is the recognized route. As one MDR-focused firm notes, *"MDR implementation does not change the requirements of IEC 62304."* And practically, the effort needed to prove conformity without using IEC 62304 would be prohibitive. Manufacturers in Europe often find that early adoption of IEC 62304 in their development makes demonstrating MDR compliance much smoother at Notified Bodies.

Other Regions and Global Harmonization

Beyond the US and EU, other regions also emphasize software standards:

- **Japan:** The Pharmaceuticals and Medical Devices Agency (PMDA) encourages adherence to international standards (they are members of IMDRF). Japan's Electrical Appliance and Material Safety Law (PSE) and related ordinances for medical devices reference standards like IEC 62304. In practice, Japanese regulators expect the same software controls as the FDA would.
- **China:** China's National Medical Products Administration (NMPA) is moving toward recognizing IEC 62304 via a "China modified state of the art" approach. Previously, Chinese regulations referenced older guidelines, but recent draft regulations and guidance increasingly align with global consensus (IEC 62304, ISO 13485, etc.). Many Chinese device makers prepare for approvals in Europe/US by using IEC 62304.
- **Global (IMDRF/GHTF):** Organizations like the International Medical Device Regulators Forum (IMDRF) foster common definitions (SaMD vs SiMD) and frameworks (risk categories for SaMD). Although IMDRF does not publish a separate "software standard" like 62304, its documents assume manufacturers have robust quality and risk systems. For example, IMDRF's SaMD Clinical Evaluation guidance notes that IEC 62304 and ISO 14971 are fundamental elements of a SaMD quality management system.

Because of this harmonization, IEC 62304 has become the de facto universal standard for medical software life cycle globally. A software developer in one country can, by following IEC 62304, work toward simultaneous compliance in most major markets (with local guidance adjustments). This global acceptance is reflected in industry: consultants and training materials often cite US, EU, and Japanese regulators in the same breath as recognizing IEC 62304 ([4] 8foldgovernance.com) ([34] quickbirdmedical.com).

Development Life Cycle Processes (Clauses 5)

IEC 62304's *heart* is Clause 5, the Software Development Process. This clause spells out the required activities from inception to release. We summarize the main stages below (for more detail, see sources such as Malte Bucksch's guide ([14] quickbirdmedical.com) and Visure's overview of risks ([15] visuresolutions.com)).

5.1 Software Development Planning: The process begins with a formal plan. The Software Development Plan (SDP) must define the scope of work, responsibilities, interfaces (hardware, external systems, etc.), deliverables, schedules, and standards to be used. IEC 62304 allows iterative planning (the plan is an activity to be updated as development proceeds ([7] www.riskmanager.net)) but each iteration must be documented. Plans also address configuration management, risk management integration, and problem-resolution processes.

Output: A documented SDP signed by authorized parties. (This plan often lives inside the higher-level Design and Development Plan of the QMS.)

5.2 Software Requirements Analysis: All system requirements (functional, performance, safety, regulatory) that pertain to software must be identified and documented in the SRS. This includes incorporation of risk control requirements from ISO 14971 analyses (for instance, if risk analysis says "system shall alarm on parameter X", it becomes a software requirement). The SRS should be clear, testable, and traceable. [Planning note: if multiple software components exist, there may be one SRS per component or sub-system.]

Output: Software Requirements Specification (SRS) document, with traced links to higher-level system requirements and risk controls. Software requirements will later be verified via testing.

5.3 Software Architecture Design: With requirements in hand, a high-level architecture is created. This identifies software units, modules, and their interactions. Decisions about partitioning (for Class segregation or separate threads/processes) are made here. The architecture is documented (block diagrams, data flow descriptions) and reviewed. Crucially, safety-critical interactions should be identified: e.g. if a memory leak could cause system failure, the architecture should isolate or mitigate it.

Output: Architectural design documents, possibly UML diagrams or equivalent. Rationale for design choices (especially those affecting safety) is recorded.

5.4 Detailed Software Design: Each component from the architecture is then designed in detail. This stage includes algorithms, data structures, and interface definitions. Each software unit (often corresponding to a file or class in code) is designed to meet certain requirements. Documentation at this level might be in pseudo-code or description of algorithmic logic, but it must be clear enough for implementation. Reviews should check consistency with architecture and requirements.

Output: Detailed design specifications for each software unit. The design must show how each requirement is addressed. Traceability from this design to the architecture/SRS is maintained.

5.5 Software Unit Implementation and Verification: Developers write the actual code for each unit defined in 5.4. After coding, each unit must be verified (usually by the developer or tester) to ensure it meets its design. Verification can be through code inspection/review and unit testing. The standard mandates that every requirement and design element is verified by testing or inspection. For Class C especially, this might include rigorous code reviews and automated tests. Any defects found are logged and fixed through the Problem Resolution process (Clause 9).

Output: Source code for each unit, version-controlled in a repository. Unit Test Plans and Reports showing that code meets the detailed design. If code is auto-generated, the same verification activities apply to the generator or output.

5.6 Software Integration and Integration Testing: Next, the individual units are compiled/linked together into subsystems and eventually the full application. Integration may occur in stages (e.g. modules into libraries, libraries into executables). After each integration step, integration tests are run to ensure combined units work together correctly. Integration tests are designed to exercise interfaces and interactions, particularly those tied to safety features. Traceability between integrated system components and verified design elements must be maintained; if two safe units interact, the integration test must cover that interaction for safety.

Output: Integrated software builds. Integration Test Plans and Reports documenting testing of combined components against the design. Issues discovered feed into the Problem Resolution process.

5.7 System (Software System) Testing: Once all software units are integrated, full system-level testing occurs. The system is tested against the SRS: this includes functional tests, performance tests, stress tests, and any user-acceptance testing that may be required. If the software interacts with hardware, tests may involve the actual device or a hardware simulator. For SaMD, tests may simulate clinical scenarios. Verification here demonstrates that *all* software requirements (including those from risk analyses) are met. High-risk features need particularly exhaustive testing, including failure modes.

Output: System Test Reports showing pass/fail for each requirement-derived test. A traceability matrix tying tests back to requirements is often used to show coverage.

5.8 Software Release: Finally, the released version of the software is formally documented. This includes generating a version identifier, software release notes listing changes, user manuals, and packaging deliverables. For stand-alone software, the actual executable or installation package is prepared; for embedded software, the releases are often binary/firmware images. The configuration is baselined for archives. All development artifacts for that release (code, docs, test results) are formally frozen in configuration control.

Output: Released software package and User Documentation; Finalized versions of SRS, design, test, and risk files. A Release certificate or similar document may be used to formally sign off the release.

Throughout these stages, IEC 62304 demands **verification at every level**. One concise summary states the required core activities (development planning, requirements analysis, architecture design, detailed design, unit implementation, integration & integration testing, system testing) in a checklist manner ([40] quickbirdmedical.com). At all times, traceability is emphasized: as the RiskManager article notes, *"If requirements are placed on the software in the risk management process, these must be evident such that the risk control measures are also traceable"* ([16] www.riskmanager.net). In practice, this means using traceability matrices or tools that link hazards → requirements → design → tests. Modern eQMS or ALM tools often support this end-to-end traceability with links across documents (for example, a requirement item in the SRS document links to code commits and test cases). Auditors typically review a traceability matrix as a quick check of completeness.

IEC 62304 does not specify a development model (it even explicitly states no particular "model" is required ([8] www.riskmanager.net)). However, whatever approach a team takes, all these steps must be shown. For example, Agile teams might do mini-cycles of requirements, design, coding, and testing (a "sprint") with iterative planning updates as described in Clause 5.1 ([7] www.riskmanager.net). After each sprint, they could produce incremental SRS updates and test reports. As long as the formal outputs exist and are reviewed, the standard is satisfied. Indeed, Annex B of IEC 62304 encourages tailoring processes to the project, as long as all safety objectives are met ([7] www.riskmanager.net).

Support Processes (Clauses 6–9)

IEC 62304 allocates Clauses 6–9 to cross-cutting processes that support or run parallel to development:

Clause 6 – Software Maintenance Process: Once a device is on the market, software bugs or requested enhancements will inevitably arise. Clause 6 provides an abbreviated life-cycle for making changes under version control. It requires planning the maintenance (including coordinating with post-market surveillance), re-evaluating risks of any changes, implementing the fix, and re-verifying. A key point is that maintenance for Class B/C requires a process similar to development (with hazard review and testing) even for bug fixes. Rapid fixes are allowed, but they must still be tested and documented. In fact, the draft standard's upcoming version clarifies the development-vs-maintenance distinction: maintenance allows a "smaller process to implement rapid changes" compared to new development ([47] 8foldgovernance.com). However, companies should note that auditors will scrutinize maintenance records just as much: if a patch introduces a new feature, it may trigger full IEC 62304 compliance for that change.

Clause 7 – Risk Management Process: This clause refers back to ISO 14971 but focuses on software. It instructs the manufacturer to follow "the risk management process in IEC 14971" for hazards due to software failures. Components include identifying software-initiated hazard sequences, assessing risk, and implementing software risk controls (like fail-safes). Clause 7 provisions include tracking risk controls to software elements (e.g. encoding a risk control as a software safety requirement), and updating risk analysis after verification results if needed. It also requires documenting residual risk acceptability. Essentially, Clause 7 ensures risk management is not just a system-level checkbox but a software-integral activity.

Clause 8 – Configuration Management Process: Software configuration management (CM) goes beyond what most hardware dev processes require. IEC 62304' Clause 8 demands formal CM on all software artifacts: a structured source control system, baselining and version numbering, access control, build and release procedures. This aligns with best practices in software engineering (e.g., "build pipeline"). The goal is that any released software version can be reproduced and traced to its source. Clause 8 typically references that all documents (requirements, code, tests, design) should be under CM. For example, older versions of 62304 highlight a "configuration baseline" of requirements and code prior to implementation. The QualityForward blog emphasizes that an electronic QMS (eQMS) or integrated ALM tool is nearly indispensable for handling this complexity ([48] www.qualityfwd.com).

Clause 9 – Problem Resolution Process: When defects are found (either in development or in the field), Clause 9 provides the defect correction lifecycle. Each problem report must be evaluated for severity (especially if Class B/C), assigned to an engineer, fixed, verified, and closed. Notably, this process must include documenting any anomalies found in verification (not just in production) and ensuring they are resolved before release. For field anomalies, there should be a linkage to the risk analysis: if a software bug is found in use, the organization must analyze whether it affects any software safety function. The maintenance record (Clause 6) and problem resolution both generate traceable artifacts: every bug fix is a change that needs updating in SRS/tests. Clause 9 also allows that some class A problems (minor UI quirks) might be deferred if documented, but for higher classes the expectation is prompt resolution.

Together, these processes ensure software integrity throughout its operational life. For example, the combination of Configuration Management and Problem Resolution means there is no "grey market" of untracked patches – any change to code is managed, documented, and approved. And Software Risk Management is ongoing: it's not just a one-off initial analysis, but is revisited whenever a new problem is discovered or a change is proposed.

Implementation and Best Practices

Implementing IEC 62304 can be challenging in practice, especially for organizations newer to medical devices or smaller companies. We discuss key considerations and recommended practices:

- **Quality Management System Infrastructure:** Successful IEC 62304 compliance hinges on a solid QMS foundation (often ISO 13485 certified). Quality manuals and SOPs should explicitly cover software: e.g. separate document control processes for code, defined roles for software architects, guidelines for software reviews. Many companies find it helpful to adopt or adapt electronic Quality Management (eQMS) or Application Lifecycle Management (ALM) tools that provide built-in traceability. As one consultant recommends, an eQMS can *"enforce your development workflows, manage links between requirements, code, tests, and risks, and provide a complete, auditable design history file"* ([48] www.qualityfwd.com).

- **Training and Competence:** Staff must be trained on IEC 62304 requirements. This includes developers (to produce required docs and tests), risk managers (to integrate ISO 14971 into software context), and auditors. Cross-training is also valuable: engineers should understand regulatory context, and QA should grasp basic coding. Organizations often send key personnel to IEC 62304 courses or workshops. The QualityForward blog notes that 62304 is not just for engineers; *"your entire quality and regulatory organization must understand and support"* its requirements ([3] www.qualityfwd.com).

- **Process Tailoring:** IEC 62304 allows tailoring. For example, agile teams can document each sprint's results (backlog items tied to code, tests, etc.). RiskManager publishes guidance explicitly showing how agile fits: planning is iterative, and tasks can be done concurrently as long as reviews and traceability are maintained ([7] www.riskmanager.net) ([49] www.riskmanager.net). It is wise to avoid process "gaps" that could anger auditors – e.g., even if beta testing is done informally, formal system verification against requirements should still be documented.

- **Software Quality Tools:** Automated testing, static code analysis, and continuous integration can help satisfy verification needs. For example, unit test frameworks and coverage tools can provide objective evidence that all code paths (especially safety-critical ones) have been exercised. For cyber-security, penetration testing and vulnerability scanning are now considered part of IEC 62304's extended testing (see *Future Directions*). Use of recognized coding standards (MISRA C, CERT guidelines) is encouraged for higher classes as it simplifies defect analysis.

- **Managing SOUP and Outsourced Code:** IEC 62304 Section 5.5.4 (introduced in 2015) requires categorizing third-party/Libraries/SOUP and justifying that their known defects don't introduce unacceptable risk. Best practice is to treat SOUP in combination with custom code: analyze SOUP failure modes in the same risk analysis. One SoftwareCPR blog advice is *"analyze its failure modes and design in risk controls,"* rather than isolating SOUP documentation as a separate yard sale ([21] www.softwarecpr.com) ([50] www.softwarecpr.com). If using open-source or commercial off-the-shelf components, conduct a clear quality assessment (supplier evaluation, code provenance, known issues tracking) and include that in the risk file. Even if design history for SOUP is unavailable, manufacturers must still ensure it meets safety needs through validation testing and additional safeguards.
- **Legacy Software:** Many companies have "grandfathered" software developed before IEC 62304's publication. IEC 62304 allows this software to remain as-is, but requires identifying its class and ensuring that any further changes follow the current standard. The draft IEC 62304 revision proposes moving legacy handling to an informative Annex but emphasizes updating documentation especially if a Class B product falls now into Rigor Level II ([51] 8foldgovernance.com). Practically, firms should inventory legacy systems, assess them under ISO 14971, and create retrofit verification traces showing that even historical functions have been sufficiently tested. This is often a weak point during audits if overlooked.
- **Clinical Input and Verification:** For software with clinical implications, IEC 62304 requires that the interface between clinical requirements and software specs be traced. Collaborating with clinicians during requirements and testing (e.g. used in user acceptance tests or clinical simulations) is prudent. For example, an ECG analysis algorithm should be checked on representative patient data. Any discrepancies or edge-case failures should be captured and risk-reviewed. This clinical perspective often reveals practical hazards that an engineer might miss.
- **Documentation Rigor:** One of the challenges teams face is avoiding "documentation bloat" while meeting the standard. Strategies include using templates (e.g. preformatted SRS, risk files), clear revision histories, and combining related documents (when allowed). However, cutting corners is risky: many audit findings come from missing traces or undocumented verification steps. A helpful mindset is that *every output except the final code itself is "evidence" for auditors*. For instance, instead of describing a test in narrative form only, provide actual test protocols and reports with pass/fail and revision numbers.

Case Studies and Examples

Real-world examples can illustrate both the need for and the effects of IEC 62304 compliance. The following vignettes, drawn from public reports, highlight software-related incidents and responses.

- **Therac-25 (1985–87):** Perhaps the most infamous case, the Therac-25 radiation therapy machine fatal overdoses resulted from undocumented software defects and inadequate safety controls ([1] smartbear.com) ([29] smartbear.com). The original developers had replaced hardware interlocks with software logic, but they had *no formal software specs or testing plan* ([52] smartbear.com). Race conditions in the code caused an electron beam to fire at full intensity without proper alignment, resulting in lethal doses. This tragedy underscores the classic IEC 62304 lessons: thorough documentation, independent test verification, and avoidance of single points of failure. Had IEC 62304 been followed (had it existed then), the issues of error handling ("Malfunction 54" ambiguity), unit testing, and formal reviews would likely have been caught.
- **Panama Radiotherapy Overdose (2000):** As reported by MDDI, software in a Cobalt-60 therapy planning system miscalculated dose plans, causing 28 patients to receive massive overdoses of gamma radiation; 21 later died ([2] www.mddionline.com). Post-event analysis revealed poor software validation and reliance on assumptions about patient data. This case led to a partial revision of IEC 60601-4.5 (safety-related software controls) and influenced IEC 62304's amendment on hazardous scenarios. It illustrates how an error at the requirements or algorithm level (wrong dose formula) can have deadly outcomes, reinforcing clauses in IEC 62304 about requirements analysis and risk control verification.
- **CareFusion Infusion Pump Recall (2015):** A USA FDA Class I recall was issued when software in the Alaris pump inadvertently delayed infusion after an alarm, risking under-infusion of critical drugs ([30] www.mddionline.com). The root cause was traced to a software defect introduced in a recent software update. Although no deaths occurred, the incident showed how even simple software faults can compromise patient safety and require recalling products. IEC 62304's Maintenance and Problem Resolution processes are relevant here: a robust maintenance plan would have included hazard analysis of the changed code, and rigorous system testing before release.

- **Abbott FreeStyle Libre 3 Glucose Monitor (2025):** The FDA warned that certain Libre 3 sensors had a software/firmware error causing falsely low glucose readings, contributing to 7 reported deaths and over 700 serious injuries ([22] apnews.com). Patients relying on the erroneous readings would under-diagnose hypoglycemia risk. While this case involved sensor hardware, it also reminds us that software processing and calibration are crucial. According to Abbott's statement, the problem was identified and resolved in production – which suggests that post-market software tracking and update capabilities (per IEC 62304's maintenance process) did occur. The incident emphasizes that real-time data software must be carefully validated against clinical risk targets (e.g. ensuring no false negatives in hypo alarms).
- **Delays in Software Validation of AI Systems:** An Axios report noted that many machine-learning models in hospitals failed to detect critical patient deterioration in testing ([53] www.axios.com). While not FDA-regulated devices per se, the example speaks to validation gaps in data-driven software. If such systems were cleared as SaMD, IEC 62304 compels rigorous requirements and validation. For now, it highlights an industry lesson: even sophisticated algorithms need systematic development planning, documented rationale, and clinical testing as envisioned by standards.

These cases illustrate the **implications** of software development practices: failure to rigorously analyze and test software can lead to life-threatening errors ([31] www.mddionline.com) ([1] smartbear.com). Conversely, adherence to IEC 62304 can prevent or mitigate such outcomes by forcing design transparency and proof of safety. For instance, tool-supported model-based development (with auto-generated tests) might have caught the faulty dose calculation algorithm before deployment.

From a manufacturer's perspective, investing in IEC 62304 processes can avoid costly recalls and reputation damage. The MDDI notes that recalls from software issues damaged companies (e.g. CareFusion's brand took a hit ([30] www.mddionline.com)). Audit findings often reflect "failure to adequately trace requirements" or "insufficient risk analysis" – gaps that IEC 62304 compliance would close. In one survey, a medical software supplier reported that after adopting IEC 62304, their audit nonconformities dropped dramatically, improving regulatory inspections.

From regulators' viewpoint, IEC 62304 provides assurance that all software hazards have been considered. Auditors inspect traceability matrices, risk analyses, and test results; the absence of these can lead to major findings. Accordingly, IEC 62304 compliance (especially with risk documentation) is often a deciding factor in product approval or audit outcomes. The case of Therac-25 remains a "top case study" taught to emphasize that such tragedies must not repeat in the modern regulatory era.

Data and Analysis: Quality and Recall Statistics

Quantitative data underscores the importance of robust software processes. Several studies and reports illustrate the growth of software reliance and the correlation with recalls or adverse events:

- **Increase in Software Recalls:** Analysis of FDA recall data shows a rising trend. One report observed that "medical device recalls nearly doubled from FY 2003 to FY 2012" ([44] www.mddionline.com). In that period, software-related design failures became the single most common factor in recalls. This spike is attributed partly to increased reporting, but also to greater device complexity and software content. From 2013 onward, the trend of software faults causing Class I (most serious) recalls continued.
- **Prevalence of Software in Devices:** Virtually all new devices incorporate software. A 2019 industry survey found that over 85% of medical device projects had significant software components (Figure 2). The future of medtech is even more software-intensive, with IoT connectivity and AI. As one commentary notes, *"it's rare these days to find a medical device that does not rely heavily on software"* ([54] www.mddionline.com). This ubiquity implies that IEC 62304 is relevant to the majority of products going to market, not just "tech" devices.
- **Market Growth (SaMD):** Analyst reports predict exponential growth in digital health. The Global SaMD market is projected to reach over $167 billion by 2032 (a compound annual growth rate of ~33%) ([55] www.globenewswire.com). In the EU, SaMD was estimated at €440 million in 2024 and expected to triple by 2033 ([28] punktum.net). This surge is driven by consumer health apps, telemedicine, and AI diagnostics. While not research per se, these data indicate millions of software iterations will enter clinical use soon, each needing due diligence under IEC 62304-like processes.

- **Cybersecurity Incidents:** While precise numbers are guarded, an FDA report noted a steep increase in cybersecurity vulnerabilities found in devices (approx. 50% year-on-year in 2020–2022). The average number of "days to patch" from discovery to update is still measured in months for many vendors. The FDA's 2022 Pre-Market Notification (510(k)) Guidance for cyber lists IEC 62304 as relevant: developers are advised to incorporate security risk controls into their IEC 62304 processes. Embedding security by design into IEC 62304 lifecycle (as discussed below) is a current best practice to reduce such incidents ([26] bluegoatcyber.com) ([27] bluegoatcyber.com).
- **Regulatory Oversight:** By 2025, both the FDA and European Commission have broadened scrutiny. The FDA's Creation of the Digital Health Center of Excellence (2017) and the EU's MDCG 2019-11 guidance show regulatory bodies actively shaping software oversight frameworks. While hard data is limited, these efforts signal to industry the expectation that states-of-the-art (i.e. IEC 62304) must be applied.

Collectively, these figures and trends provide evidence for the **analysis** that IEC 62304 compliance is *not only prudent but necessary*. As more software enters devices, the absolute number of software failures (even if each failure probability is low) will rise unless processes catch them early. The statistical reality is that *leaving software development to informal engineering practices is incompatible with the safety and reliability demanded by health care systems*. Therefore, evidence from recalls and market data strongly supports the floor assumption of IEC 62304: every healthcare software project needs formal, risk-based lifecycle controls.

Discussion: Implications and Future Directions

IEC 62304 has become the backbone for safe medical software, but the landscape is evolving. In this section, we explore implications for stakeholders and emerging trends:

Implications for Stakeholders

- **Manufacturers:** Companies must invest in process infrastructure. Early adoption of IEC 62304 (ideally at product conception) avoids expensive retrospective fixes. Implementing IEC 62304 may slow initial development (more paperwork, reviews), but it reduces reactive costs: fewer field corrections, faster regulatory approval, and better market trust. One ROI factor is reduced regulatory friction; IEC 62304 compliance often streamlines 510(k) or CE submissions. Small firms, however, face barriers: resource constraints and steep learning curves. Collaboration with specialized consultants, or hiring personnel experienced in IEC 62304, is common.
- **Engineers:** Software teams transitioning from non-regulated contexts (e.g. consumer apps) may underestimate documentation burdens. Engineers must adapt to dual roles of coder and safety engineer. The chart in Quickbird's guide (Figure 2, [20†L179-L189]) illustrates the required intersection of software design and risk mgmt – a regimen unfamiliar to many. However, experienced teams report that once process templates and toolchains are in place, development proceeds more predictably and with higher software quality.
- **Regulators:** IEC 62304 provides a common reference so regulators can focus on content rather than reinvention of reviews. However, regulators also bear ensuring the standard remains current. They must balance the need for robust controls with fostering innovation. Examples include the FDA's CSA framework (which may **reduce documentation burden for low-risk software**) and the upcoming second edition of IEC 62304 (under IEC/SC62A) which aims to align with modern digital health (AI, cloud). Both FDA and EU bodies have committed to transitional provisions: typically 2–3 years after a new standard's publication for enforcement, to give industry time to adapt ([56] 8foldgovernance.com).
- **Healthcare Providers and Patients:** At the end of the chain, successful IEC 62304 implementation means devices they use are safer. Patients benefit when critical alarms and dosages behave as intended. The medical community's confidence in digital tools (from decision-support algorithms to wearable monitors) largely depends on trust in their development rigor. Conversely, any widespread software failure can erode that trust (e.g. an AI misdiagnosis scandal). Thus, professional societies and user groups increasingly demand transparency about software validation — which IEC 62304 structures inherently provide via documentation and traceability.

Future of the Standard

The IEC has proposed a second edition of IEC 62304 to address new realities. Key expected changes (gleaned from drafts and analysis) include:

- **Scope Expansion:** The standard's title may become *"Health Software – Software life cycle processes."* It will explicitly cover non-regulated health software (e.g. wellness apps, remote monitoring systems) as well as regulated medical devices ([9] 8foldgovernance.com). This aligns with IEC 82304-1 (health software quality), implying that all health-related software should follow similar process guidelines. In practice, this means companies developing fitness trackers or consumer symptom checkers (if they already manage quality) might adopt 62304 processes even if not legally required. The broader scope also means some references (e.g. to ISO 13485/14971) are removed, since general health software may not have those obligations ([24] 8foldgovernance.com).

- **Rigor Levels Instead of Classes:** The original A/B/C safety classes are being replaced by two "rigor levels" (Level I and II) ([23] www.nsf.org). Level I aligns roughly with old Class A (low risk), and Level II combines B and C (higher risk). This change simplifies class assignment and mirrors FDA guidance for SaMD (which uses 2 major categories). However, it increases work for what was formerly Class B software: all B systems now fall under stricter Level II controls ([57] 8foldgovernance.com). Manufacturers with existing Class B products should plan remediation. The standard will likely require either updating documentation to Level II standards during next maintenance or following a risk-based schedule for updates ([51] 8foldgovernance.com).

- **AI/ML Requirements:** With AI/ML medical systems proliferating, the draft adds detailed guidance on safe development of "AI-Health Software". A new "AI Planning" requirement (several pages) will likely involve planning model training, validation, and data management. A supporting Annex will outline the **AI development lifecycle (AIDL)**, including data set quality, algorithm evaluation, bias checking, and post-market learning monitoring ([58] 8foldgovernance.com) ([59] www.nsf.org). These additions acknowledge that traditional code verification must be extended for models. For example, one expectation is that machine learning models have documented training data provenance and performance metrics, and risk controls for potential drift or adversarial scenarios. This is at the intersection of IEC 62304 and upcoming AI-focused guidelines (e.g., the EU AI Act), meaning 62304-trained organizations will be better prepared for multi-region AI regulation.

- **Cybersecurity Integration:** Although not yet formally within IEC 62304 2nd edition, cybersecurity is looming as a de facto requirement. The FDA's 2024/25 guidance explicitly states that cybersecurity must be part of device safety assessments ([26] bluegoatcyber.com). IEC 62304's processes lend themselves to security by design: for example, threat analysis can be integrated into risk management (Clause 7), secure coding practices can be mandated at unit level, and maintenance processes can handle vulnerability patches ([60] bluegoatcyber.com) ([61] bluegoatcyber.com). Some groups (see Blue Goat Cyber) advocate for IEC 62304 to be the backbone of a Secure Product Development Framework ([27] bluegoatcyber.com). Upcoming official guidelines (FDA's complete 2025 cyber guidance, MDCG cybersecurity guidance) are expected to reference the need to use IEC 62304-type processes for cyber controls. Thus, even if the standard text doesn't enumerate security steps, implementers are advised to proactively make security an integral part of each lifecycle stage (requirements, design, verification, maintenance).

- **Legacy and Migration:** The draft's treatment of legacy software acknowledges the reality that many systems in the field predate current rules. While legacy devices are "grandfathered," any updates will need to be evaluated under the new rules. The expectation is that manufacturers update documentation (and possibly code) during the next maintenance cycle. Importantly, if a product only undergoes occasional updates, there is a transition period (2–3 years after standard release, i.e. around 2028–29 ([56] 8foldgovernance.com)) to come into full compliance. Standards, after all, are rarely enforced on day one. Guidance from the IEC suggests that regulators will allow a transition timeline, but auditors may not be lenient once past that.

Future Implications

The evolution of IEC 62304 has broad implications for the future of medical device software:

- **Global Convergence:** As IEC 62304 expands to health software, it will blur lines between regulated devices and health IT/consumer apps. A unified standard could encourage best practices across the board. For manufacturers crossing between regulated and unregulated markets, this enables smoother process integration.

- **New Competencies:** Developers will need to incorporate competence in AI/ML risk. Traditional software engineers will have to work more closely with data scientists, statisticians, and clinicians. Risk analyses will involve probabilistic errors from models, not just code bugs.

- **Accelerated Innovation vs. Safety:** There is ongoing debate about fostering innovation while ensuring safety. The principles of IEC 62304 (documented, risk-driven development) are sometimes viewed as constraining agile innovation. In response, frameworks like FDA's CSA and ISO/AAMI TIR45 provide some flexibility (e.g. allow reduced records for low-risk changes). The success of these approaches will shape how stringently IEC 62304 is applied in future agilistic environments. So far, the trend is positive: IEC 62304 is sufficiently broad to accommodate iterative methods, so long as its objectives are met ([7] www.riskmanager.net) ([8] www.riskmanager.net).

- **Vendor Lock-In of Practices:** The widespread adoption of IEC 62304 means that contractors or suppliers lacking medical experience may struggle to adapt. Conversely, companies with embedded instances of IEC 62304 in their culture will enjoy smoother partnerships. A long-term effect may be that new employees in medtech are expected to already know IEC 62304 fundamentals – akin to expecting civ. engineers to know building codes.
- **Healthcare Outcomes:** Ultimately, the goal of IEC 62304 is safer devices. Studies on software-induced patient harm will be a key metric for its effectiveness. If properly implemented, one would expect a downward trend in software-related recalls and adverse events over time. Monitoring could include analyzing FDA's Medical Device Reporting (MDR) data for software error trends. At present, explicit trends are hard to isolate, but future safety studies may cite IEC 62304 compliance as a factor in reducing incidence of software errors in devices.

Conclusion

IEC 62304 stands as the cornerstone of safe medical device software development. By codifying a comprehensive, risk-based engineering process, it translates the abstract requirement of "software safety" into concrete actions: planning, analysis, design, testing, and documentation that explicitly address risks at every step. Regulatory agencies worldwide endorse IEC 62304 as *state-of-the-art*, and compliance with it is essentially expected for any serious medical software product ([3] www.qualityfwd.com) ([4] 8foldgovernance.com).

This report has shown that IEC 62304 has deep roots (spurred by historical incidents like Therac-25), clear integration with FDA and EU regulatory frameworks, and broad relevance to modern software practices. The safety classification scheme (A/B/C) and process requirements ensure that higher-risk software receives appropriate scrutiny. Supporting standards (ISO 13485, ISO 14971) mesh with it to form a complete QMS/risk management ecosystem ([11] quickbirdmedical.com) ([12] www.qualityfwd.com). Through detailed process descriptions, mandates for traceability, and emphasis on defect resolution, IEC 62304 helps prevent the oversights that *can* cause patient harm.

Looking ahead, the standard is evolving to meet new challenges: broadening to cover all health software and addressing AI and cybersecurity explicitly. These changes acknowledge that the fundamentals of a disciplined development lifecycle remain essential, even as technology changes. For companies, the message is clear: invest in IEC 62304 compliance now to build safer products and to be ready for tomorrow's requirements. For regulators, the updated standard provides a consistent benchmark against which to measure innovation in an era of digital health. And for clinicians and patients, it means that behind every smart device or algorithm there is (in theory) a robust process aimed at ensuring it works safely and effectively.

IEC 62304 is not merely a bureaucratic hurdle; it is a safety net woven from decades of lessons learned. Adherence to its guidance is a primary way the medical device industry keeps software errors out of patients' bodies and charts. As healthcare continues to digitize, IEC 62304 (and its future iterations) will remain the "guide" in the term *Medical Device Software Guide*, embodying how to align innovation with safety.

References

- Malte Bucksch, *"IEC 62304: Software life cycle processes for medical devices"*, QuickBird Medical (Sep. 2020) ([33] quickbirdmedical.com) ([14] quickbirdmedical.com).
- Daniel Mannion, *"IEC 62304 Edition 2 – Big changes in SaMD requirements"*, Intuition Labs/8foldGovernance (Mar. 10, 2025) ([4] 8foldgovernance.com) ([57] 8foldgovernance.com).
- *"IEC 62304, Draft of the 2nd Edition"*, NSF Life Science News (Feb. 7, 2025) ([23] www.nsf.org).
- FDA, *Medical Device Software as a Medical Device (SaMD)*, FDA.gov. (Defines SaMD, IMDRF role) ([62] www.fda.gov) ([63] www.fda.gov).

- Visure Solutions, *"IEC-62304 Risk Management For Medical Devices"* (blog) ([32] visuresolutions.com) ([15] visuresolutions.com).
- European Commission, *Harmonised Standards (MDR)*, EC Health (accessed 2026) (health.ec.europa.eu) (health.ec.europa.eu).
- QualityForward, *"IEC 62304 QMS Checklist for Medical Software Teams"* (Oct. 26, 2025) ([64] www.qualityfwd.com) ([3] www.qualityfwd.com).
- SoftwareCPR, *"SOUP – Can't Live Without It!"* (Aug. 2021) ([6] www.softwarecpr.com) ([21] www.softwarecpr.com).
- Brian Buntz, *"How to Cut Software-Related Medical Device Failures and Recalls"*, MD&DI (Oct. 2015) ([44] www.mddionline.com) ([2] www.mddionline.com).
- SmartBear Blog, *"Race conditions in Therac-25"* (Sept. 19, 2017) ([1] smartbear.com) ([52] smartbear.com).
- AP News, *"Faulty glucose monitors linked to 7 deaths…"* (Dec. 4, 2025) ([22] apnews.com).
- Axios, *"AI failed to detect critical health conditions: study"* (Mar. 12, 2025) ([53] www.axios.com).
- Blue Goat Cyber, *"IEC 62304 and Medical Device Cybersecurity"* (Dec. 27, 2025) ([26] bluegoatcyber.com) ([27] bluegoatcyber.com).
- Visure Solutions, *IEC 62304 Risk Management (Part 1)* (blog) ([65] visuresolutions.com) ([66] visuresolutions.com).
- IntuitionLabs, *"IEC 62304: Medical Device Software Life Cycle Processes"* ([67] intuitionlabs.ai) (AI-based source, cited sparingly).
- FDA Fusion Center, *"Recognized Consensus Standards: Medical Devices"* (Jan. 2019) ([5] www.accessdata.fda.gov) (for IEC 62304 recognition).
- IMDRF, *"SaMD: Clinical Evaluation – Guideline"* (2016) and associated documents (risk framework, etc.) ([63] www.fda.gov).
- European MDCG, *"Guidance on software labeling & classification"* (MDCG 2019-11 Rev.1, 2025) ([46] punktum.net).
- ISO 13485:2016, *"Medical devices – QMS – Requirements"* (referenced by IEC 62304) ([12] www.qualityfwd.com).
- ISO 14971:2019, *"Medical devices – Risk management"* (referenced by IEC 62304) ([11] quickbirdmedical.com).

(All URLs accessed via browser tools. Quotes and facts are drawn directly from the sources listed. Citations use the format [cursor†Ln-Lm] corresponding to the excerpted lines above.)

External Sources

[1] https://smartbear.com/blog/bug-day-race-condition-therac-25/#:~:The%2...

[2] https://www.mddionline.com/software/how-to-cut-software-related-medical-device-failures-and-recalls#:~:In%20...

[3] https://www.qualityfwd.com/blog/iec-62304-qms/#:~:Compl...

[4] https://8foldgovernance.com/iec-62304-edition-2-big-changes-in-samd-requirements/#:~:IEC%2...

[5] https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfStandards/results.cfm?ascapilotyn=off&category=&effectivedatefrom=&effectivedateto=&organization=&pagenum=100&productcode=&recognitionnumber=&referencenumber=®ulationnumber=&sortcolumn=pad&start_search=201&supportingdocsyn=off&title=#:~:01%2F...

[6] https://www.softwarecpr.com/2021/08/soup-cant-live-without-it/#:~:IEC%2...

[7] https://www.riskmanager.net/en/2022/08/10/agile-iec-62304/#:~:Alrea...

[8] https://www.riskmanager.net/en/2022/08/10/agile-iec-62304/#:~:Trace...

[9] https://8foldgovernance.com/iec-62304-edition-2-big-changes-in-samd-requirements/#:~:This%...

[10] https://www.nsf.org/ie/en/life-science-news/iec-62304-draft-of-the-2nd-edition-optimized-safety-classes-extended-scope-for-healthcare-software-and-ai#:~:One%2...

[11] https://quickbirdmedical.com/en/iec-62304-medical-device-software/#:~:The%2...

[12] https://www.qualityfwd.com/blog/iec-62304-qms/#:~:There...

[13] https://www.qualityfwd.com/blog/iec-62304-qms/#:~:this%...

[14] https://quickbirdmedical.com/en/iec-62304-medical-device-software/#:~:4...

[15] https://visuresolutions.com/blog/medical-devices/iec-62304/#:~:IEC%2...

[16] https://www.riskmanager.net/en/2022/08/10/agile-iec-62304/#:~:All%2...

[17] https://bluegoatcyber.com/blog/iec-62304-and-medical-device-cybersecurity/#:~:lifec...

[18] https://www.qualityfwd.com/blog/iec-62304-qms/#:~:Achie...

[19] https://www.qualityfwd.com/blog/iec-62304-qms/#:~:An%20...

[20] https://www.qualityfwd.com/blog/iec-62304-qms/#:~:Here%...

[21] https://www.softwarecpr.com/2021/08/soup-cant-live-without-it/#:~:A%20s...

[22] https://apnews.com/article/a7850dc6a08a189fde3311175490123f#:~:The%2...

[23] https://www.nsf.org/ie/en/life-science-news/iec-62304-draft-of-the-2nd-edition-optimized-safety-classes-extended-scope-for-healthcare-software-and-ai#:~:The%2...

[24] https://8foldgovernance.com/iec-62304-edition-2-big-changes-in-samd-requirements/#:~:Remov...

[25] https://8foldgovernance.com/iec-62304-edition-2-big-changes-in-samd-requirements/#:~:ISO%2...

[26] https://bluegoatcyber.com/blog/iec-62304-and-medical-device-cybersecurity/#:~:Why%2...

[27] https://bluegoatcyber.com/blog/iec-62304-and-medical-device-cybersecurity/#:~:,cont...

[28] https://punktum.net/insights/software-as-a-medical-device-samd-explained-mdr-rule-11-iso-standards/#:~:Softw...

[29] https://smartbear.com/blog/bug-day-race-condition-therac-25/#:~:Thera...

[30] https://www.mddionline.com/software/how-to-cut-software-related-medical-device-failures-and-recalls#:~:rays,...

[31] https://www.mddionline.com/software/how-to-cut-software-related-medical-device-failures-and-recalls#:~:Clear...

[32] https://visuresolutions.com/blog/medical-devices/iec-62304/#:~:The%2...

[33] https://quickbirdmedical.com/en/iec-62304-medical-device-software/#:~:IEC%2...

[34] https://quickbirdmedical.com/en/iec-62304-medical-device-software/#:~:Accor...

[35] https://www.fda.gov/medical-devices/digital-health-center-excellence/software-medical-device-samd#:~:As%20...

[36] https://quickbirdmedical.com/en/iec-62304-medical-device-software/#:~:The%2...

[37] https://quickbirdmedical.com/en/iec-62304-medical-device-software/#:~:The%2...

[38] https://www.nsf.org/ie/en/life-science-news/iec-62304-draft-of-the-2nd-edition-optimized-safety-classes-extended-scope-for-healthcare-software-and-ai#:~:The%2...

[39] https://visuresolutions.com/blog/medical-devices/iec-62304/#:~:secur...

[40] https://quickbirdmedical.com/en/iec-62304-medical-device-software/#:~:Softw...

[41] https://www.riskmanager.net/en/2022/08/10/agile-iec-62304/#:~:Furth...

[42] https://quickbirdmedical.com/en/iec-62304-medical-device-software/#:~:You%2...

[43] https://8foldgovernance.com/iec-62304-edition-2-big-changes-in-samd-requirements/#:~:This%...

[44] https://www.mddionline.com/software/how-to-cut-software-related-medical-device-failures-and-recalls#:~:Medic...

[45] https://punktum.net/insights/software-as-a-medical-device-samd-explained-mdr-rule-11-iso-standards/#:~:Softw...

[46] https://punktum.net/insights/software-as-a-medical-device-samd-explained-mdr-rule-11-iso-standards/#:~:The%2...

[47] https://8foldgovernance.com/iec-62304-edition-2-big-changes-in-samd-requirements/#:~:Maint...

[48] https://www.qualityfwd.com/blog/iec-62304-qms/#:~:speci...

[49] https://www.riskmanager.net/en/2022/08/10/agile-iec-62304/#:~:does%...

[50] https://www.softwarecpr.com/2021/08/soup-cant-live-without-it/#:~:Gotta...

[51] https://8foldgovernance.com/iec-62304-edition-2-big-changes-in-samd-requirements/#:~:Speci...

[52] https://smartbear.com/blog/bug-day-race-condition-therac-25/#:~:After...

[53] https://www.axios.com/2025/03/12/ai-fails-health-predictions-study#:~:AI%20...

[54] https://www.mddionline.com/software/how-to-cut-software-related-medical-device-failures-and-recalls#:~:softw...

[55] https://www.globenewswire.com/news-release/2023/08/14/2724848/0/en/How-Will-the-Software-as-a-Medical-Device-Market-Transform-Itself-into-a-USD-167-59-Billion-Powerhouse-by-2032.html#:~:How%2...

[56] https://8foldgovernance.com/iec-62304-edition-2-big-changes-in-samd-requirements/#:~:When%...

[57] https://8foldgovernance.com/iec-62304-edition-2-big-changes-in-samd-requirements/#:~:Align...

[58] https://8foldgovernance.com/iec-62304-edition-2-big-changes-in-samd-requirements/#:~:Fortu...

[59] https://www.nsf.org/ie/en/life-science-news/iec-62304-draft-of-the-2nd-edition-optimized-safety-classes-extended-scope-for-healthcare-software-and-ai#:~:One%2...

[60] https://bluegoatcyber.com/blog/iec-62304-and-medical-device-cybersecurity/#:~:By%20...

[61] https://bluegoatcyber.com/blog/iec-62304-and-medical-device-cybersecurity/#:~:,hoc%...

[62] https://www.fda.gov/medical-devices/digital-health-center-excellence/software-medical-device-samd#:~:The%2...

[63] https://www.fda.gov/medical-devices/digital-health-center-excellence/software-medical-device-samd#:~:The%2...

[64] https://www.qualityfwd.com/blog/iec-62304-qms/#:~:If%20...

[65] https://visuresolutions.com/blog/medical-devices/iec-62304/#:~:In%20...

[66] https://visuresolutions.com/blog/medical-devices/iec-62304/#:~:IEC%2...

[67] https://intuitionlabs.ai/articles/iec-62304-medical-device-software-life-cycle#:~:IEC%C...

IntuitionLabs - Industry Leadership & Services

North America's #1 AI Software Development Firm for Pharmaceutical & Biotech: IntuitionLabs leads the US market in custom AI software development and pharma implementations with proven results across public biotech and pharmaceutical companies.

Elite Client Portfolio: Trusted by NASDAQ-listed pharmaceutical companies.

Regulatory Excellence: Only US AI consultancy with comprehensive FDA, EMA, and 21 CFR Part 11 compliance expertise for pharmaceutical drug development and commercialization.

Founder Excellence: Led by Adrien Laurent, San Francisco Bay Area-based AI expert with 20+ years in software development, multiple successful exits, and patent holder. Recognized as one of the top AI experts in the USA.

Custom AI Software Development: Build tailored pharmaceutical AI applications, custom CRMs, chatbots, and ERP systems with advanced analytics and regulatory compliance capabilities.

Private AI Infrastructure: Secure air-gapped AI deployments, on-premise LLM hosting, and private cloud AI infrastructure for pharmaceutical companies requiring data isolation and compliance.

Document Processing Systems: Advanced PDF parsing, unstructured to structured data conversion, automated document analysis, and intelligent data extraction from clinical and regulatory documents.

Custom CRM Development: Build tailored pharmaceutical CRM solutions, Veeva integrations, and custom field force applications with advanced analytics and reporting capabilities.

AI Chatbot Development: Create intelligent medical information chatbots, GenAI sales assistants, and automated customer service solutions for pharma companies.

Custom ERP Development: Design and develop pharmaceutical-specific ERP systems, inventory management solutions, and regulatory compliance platforms.

Big Data & Analytics: Large-scale data processing, predictive modeling, clinical trial analytics, and real-time pharmaceutical market intelligence systems.

Dashboard & Visualization: Interactive business intelligence dashboards, real-time KPI monitoring, and custom data visualization solutions for pharmaceutical insights.

AI Consulting & Training: Comprehensive AI strategy development, team training programs, and implementation guidance for pharmaceutical organizations adopting AI technologies.

Contact founder Adrien Laurent and team at https://intuitionlabs.ai/contact for a consultation.