Developing high-quality software is tough. ECLAIR is designed to help development, QA, and safety teams reach their quality goals

Coverage of IEC 62304, IEC 81001–5–1 and FDA's *General Principles of Software Validation*

1 Introduction to IEC 62304:2006

IEC 62304:2006, "Medical device software — Software life cycle processes," is an international standard issued by IEC [9]. It has been followed, in 2015, by IEC 62304:2006/Amd 1:2015, which added requirements to deal with *legacy software* and updated the software safety classification to include a risk-based approach [11].[1] IEC 62304 defines the life cycle requirements, including development and maintenance, for medical device software. The required processes, tasks, and activities are dependent on whether and how the device software can contribute to an hazardous situation: this takes into account the risk for the patients, the caregivers, the property, and the environment. Software systems are partitioned into three safety classes:[2]

Class A: the *software system* cannot contribute to a *hazardous situation*, OR, the *software system* can contribute to a *hazardous situation* which does not result in unacceptable *risk* after consideration of *risk control* measures external to the software system;

Class B: the *software system* can contribute to a *hazardous situation* which results in unacceptable *risk* after consideration of *risk control* measures external to the *software system* and the resulting possible *harm* is non-*serious injury*;

Class C: the *software system* can contribute to a *hazardous situation* which results in unacceptable *risk* after consideration of *risk control* measures external to the *software system* and the resulting possible *harm* is death or *serious injury*.

[1] This standard was last reviewed and confirmed in 2021.

[2] Emphasized expressions are given a precise meaning in [11].

2 Introduction to FDA's *General Principles of Software Validation*

The *Food and Drug Administration* (FDA) published, in 2002, its "General Principles of Software Validation" (GPSV) [7]. The GPSV document provides general validation principles that the agency considers to be applicable to the validation of medical device software or the validation of software used to design, develop, or manufacture medical devices. The final guidance document of 2002, Version 2.0, supersedes the draft document with the same title, Version 1.1, dated June 9, 1997.

GPSV's validation requirements apply to [7]:

- software used as a component, part, or accessory of a medical device;
- software that is itself a medical device (e.g., blood establishment software);
- software used in the production of a device (e.g., programmable logic controllers in manufacturing equipment); and
- software used in implementation of the device manufacturer's quality system (e.g., software that records and maintains the device history record).

In addition to the GPSV guidance on software validation, FDA cybersecurity guidance increasingly emphasizes the need for an *SPDF* (Secure Product Development Framework), i.e., documented processes and controls ensuring cybersecurity is addressed throughout the product life cycle [8]. IEC 81001–5–1 provides a structured set of security life cycle activities that can be used to support such expectations.

3 Introduction to IEC 81001-5-1

IEC 81001–5–1, "Health software and health IT systems safety, effectiveness and security — Part 5–1: Security — Activities in the product life cycle," is an international standard issued by IEC [12, 13]. It defines a set of security activities to be performed throughout the product life cycle of health software and health IT systems.

Whereas IEC 62304 focuses on software life cycle processes primarily aimed at safety, IEC 81001–5–1 complements it by defining security-specific processes and work products, including activities for: security risk management, security requirements, secure design and implementation, verification with a security focus, and post-market vulnerability handling.

The standard is commonly used to support the demonstration that an organization follows a *Secure Product Development Framework* (SPDF), i.e., a structured set of processes and controls ensuring that cybersecurity is addressed throughout the product life cycle [8].

4 Role of ECLAIR in Compliance with IEC 62304, IEC 81001-5-1 and FDA's GPSV

The ECLAIR Software Verification Platform can be used to comply with several of the requirements of IEC 62304 [11], of IEC 81001–5–1 [12, 13], and of the GPSV [7]. In particular, ECLAIR supports requirements-based development and verification, systematic defect prevention through coding standards and static analysis, objective measurement of software quality, and the enforcement of architectural constraints and segregation.

In addition, ECLAIR certification by TÜV SÜD and the entire ECLAIR functional safety ecosystem described in Section 8 below greatly simplifies obtaining all the confidence-building evidence that is required to make a solid argument justifying the use of ECLAIR in regulated projects.

5 ECLAIR Coverage of IEC 62304 and GPSV Requirements

Table 1 has been derived from Table A.1 of IEC 62304:2006, which is informative. The original table summarizes which software safety classes are assigned to each requirement. The normative section identifies the software safety classes for each requirement.

Table 2 summarizes a set of security life cycle activities addressed by IEC 81001–5–1. Tables 3 and 4 summarize the software validation principles, activities and tasks of the GPSV.

In all tables the 'ECLAIR' column indicates where ECLAIR, suitably instantiated with the appropriate package, can be used to ensure compliance or to facilitate the achievement of compliance. As ECLAIR provides direct support for MISRA guidelines as well as guidelines from other coding standards, a reference for a guideline should be taken as a reference to the corresponding *ECLAIR service* as described in the *ECLAIR User's Manual*. For example, "MISRA C:2025 Directive 3.1" corresponds to the ECLAIR service `MC4.D3.1`, "MISRA C++:2023 Rule 9.4.1" corresponds to the ECLAIR service `MP2.9.4.1` and "BARR-C:2018 Rule 4.1.a" corresponds to the ECLAIR service `NC3.4.1.a`. For ECLAIR services that do not correspond to published coding standards, the service name is given in teletype font: for example, `B.INDEPENDENCE` is the name of an ECLAIR service that supports automatically enforcing software architectural constraints [1]. A complete definition of all ECLAIR services is contained in the *ECLAIR User's Manual* and, where applicable, in the corresponding coding standard documentation referenced therein.

5.1 MISRA C:2025

MISRA C:2025 [15] is the software development C subset developed by MISRA that is a de facto standard for safety-, life-, security-, and mission-critical embedded applications in many industries, including aerospace, railway, medical, telecommunications and others. MISRA C:2025, which allows coding MISRA-compliant applications in subsets of C90, C99, C11 and C18, is supported, along with all previous versions of MISRA C, by the ECLAIR package called "MC".

5.2 MISRA C++:2023

MISRA C++:2023 [14] is the software development C++ subset developed by MISRA, which is a de facto standard for safety-, life-, and mission-critical embedded applications in many industries including aerospace, railway, medical, telecommunications and others. MISRA C++:2023 completely supersedes MISRA C++:2008 [16], the previous edition of the coding standard, which is still used by many legacy projects. MISRA C++:2023 and MISRA C++:2008 are supported by the ECLAIR package called "MP".

5.3 BARR-C:2018

The *Barr Group's Embedded C Coding Standard*, BARR-C:2018 [4], is, for coding standards used by the embedded system industry, second only in popularity to MISRA C. BARR-C:2018 guidelines include 64 guidelines dealing with language subsetting and project management as well as 79 guidelines concerning programming style. For projects in which a MISRA compliance requirement is not (yet) present, the adoption of BARR-C:2018 is a major improvement with respect to the situation where no coding standards and no static analysis is used. The adoption of the stylistic subset of BARR-C:2018 (79 out of 143 rules) can be part of complying with the MISRA requirement that a consistent programming style is adopted and systematically used as part of the software development process. Moreover, complying with BARR-C:2018, besides avoiding many dangerous bugs, entails compliance with a non-negligible subset of MISRA C:2012 [2]. ECLAIR support for BARR-C:2018 has no equals on the market: it is included in all ECLAIR packages, including the affordable package "B".

Table 1: Derived from IEC 62304 Table A.1 — Summary of requirements by software safety class

Requirement		Class			ECLAIR
		A	B	C	
5.1.1	Software development plan	X	X	X	✓[a]
5.1.4	Software development standards, methods and tools planning			X	✓[a]
5.1.5	Software integration and integration testing planning		X	X	✓[a]
5.1.6	Software verification planning	X	X	X	✓[a]
5.1.8	Documentation planning	X	X	X	✓[a]
5.1.9	Software configuration management planning	X	X	X	✓[a]
5.1.10	Supporting items to be controlled		X	X	✓[a]
5.1.11	Software configuration item control before verification		X	X	✓[a]
5.2.1	Define and document software requirements from system requirements	X	X	X	✓[b]
5.2.2	Software requirements content	X	X	X	✓[b]
5.2.3	Include risk control measures in software requirements		X	X	✓[b]
5.3.2	Develop an architecture for the interfaces of software items		X	X	✓[c]
5.3.5	Identify segregation necessary for risk control			X	✓[c]
5.3.6	Verify software architecture (points a and b)		X	X	✓[c]
5.5.2	Establish software unit verification process		X	X	✓[a]
5.5.3	Software unit acceptance criteria		X	X	✓[d]
5.5.4	Additional software unit acceptance criteria		X	X	✓[d]
5.5.5	Software unit verification		X	X	✓[d]
5.6.2	Verify software integration		X	X	✓[d]

[a] ECLAIR and its documentation can be used in the creation of a software development life cycle based on requirements, traceability, coding standards, software metrics, and static analysis.

[b] ECLAIR service `B.REQMAN` allows ensuring that all code is forward and backward traceable to documented requirements, including functional and capability requirements, requirements on inputs, outputs and interfaces, safety requirements and security requirements. `B.REQMAN` also allows tracing code to the tests and back. The integrated requirements management tool makes ECLAIR a cost-effective, complete solution for requirements-based development and testing.

[c] The *ECLAIR Independence Checker* (service `B.INDEPENDENCE`) allows enforcing constraints about the software architecture and ensuring the interface of software items are respected. For software items executing on the same processor, `B.INDEPENDENCE` can ensure that segregation is effective at the source code level.

[d] ECLAIR can be used to verify the compliance of source code to coding standards, such as the MISRA coding standards and BARR-C:2018. In turn, compliance with such standards ensures that the semantics of the source code is well defined and that certain classes of run-time errors cannot occur. ECLAIR can also be used to verify that the value of software metrics are inside prescribed ranges.

Table 2: Summary of ECLAIR coverage of IEC 81001–5–1 security life cycle activities

Activity (IEC 81001–5–1)		ECLAIR
S1	Security planning and definition of security life cycle activities	✓[a]
S2	Security requirements, including security risk controls and traceability	✓[b]
S3	Secure architecture and interface control (including segregation constraints)	✓[c]
S4	Secure implementation, defect prevention, coding rules and metrics	✓[d]
S5	Security-focused verification (including regression and change impact)	✓[e]
S6	Use of third-party components and control of what is actually used	✓[f]
S7	Support to vulnerability handling workflows through systematic detection and repeatable evidence	✓[g]

[a] ECLAIR and its documentation can be used as a foundation for a secure development life cycle based on requirements, traceability, coding standards, software metrics, and static analysis; the platform is fully scriptable and amenable to automation and CI/CD integration.

[b] ECLAIR service `B.REQMAN` supports forward and backward traceability between security requirements, implementation, and tests. This includes requirements on interfaces and constraints, as well as security risk control measures.

[c] The *ECLAIR Independence Checker* (service `B.INDEPENDENCE`) supports the formal specification and systematic checking of architectural constraints, layering, and allowed interactions, preventing bypassing of interfaces and supporting segregation arguments at source code level.

[d] ECLAIR supports compliance with coding standards (e.g., MISRA C, MISRA C++ and BARR-C) and checks numerous defect classes and software metrics, thereby preventing vulnerabilities and safety-critical defects from being introduced.

[e] ECLAIR intercepts the actual build process and analyzes the resulting system as built. Differential reports and trend analysis can be used to assess change impact and to support regression verification over time.

[f] When software is assembled from existing components (e.g., libraries), the ECLAIR service `B.SCOUT` supports precise identification of the parts that are actually used, thereby focusing verification and validation effort where it matters and supporting SOUP/open-source component assessment.

[g] Findings produced by static analysis, compliance checking and architectural constraint checking provide objective evidence that can be integrated into vulnerability management workflows, supporting triage, remediation, and verification of fixes.

Table 3: Summary of ECLAIR coverage of GPSV's requirements: Section 4

Requirement		ECLAIR
4.1	Requirements	✓[a]
4.2	Defect prevention	✓[b]
4.3	Time and Effort	✓[b]
4.4	Software Life Cycle	✓[b]
4.5	Plans	✓[b]
4.6	Procedures	✓[b,c]
4.7	Software validation after a change	✓[d]
4.8	Validation Coverage	✓[e]
4.9	Independence of review	✓[f]
4.10	Flexibility and responsibility	✓[g]

[a] ECLAIR service `B.REQMAN` allows ensuring that all code is forward and backward traceable to documented requirements, including functional and capability requirements, requirements on inputs, outputs and interfaces, safety requirements and security requirements. `B.REQMAN` also allows tracing code to the tests and back. The integrated requirements management tool makes ECLAIR a cost-effective, complete solution for requirements-based development and testing.

[b] ECLAIR and its documentation can be used in the creation of a software development life cycle based on requirements, traceability, coding standards, software metrics, and static analysis that prevents the introduction of defects before testing is reached.

[c] ECLAIR is fully scriptable and particularly amenable to the creation of automated procedures.

[d] ECLAIR, unless specifically instructed to do so, performs an analysis of the software system by intercepting the exact sequence of commands that are used to produce the complete system, therefore any change will be assessed in its global impact on the entire software system. In addition, it is possible to produce differential reports with respect to a previous analysis performed on the software, thereby obtaining an assesment of the overall impact of any changes made to the software.

[e] ECLAIR supports, among many others, several software complexity metrics that, together with the evaluation of the risk associated to each software component, allow for the rational selection of the validation activities to be applied to that component.

[f] Assistance of a high-quality tool like ECLAIR checking compliance with, e.g., the MISRA guidelines, allows to partly implement the principle of independence of review (the tool can be considered a "third party" with respect to the development team).

[g] ECLAIR's unique features allow for tailoring the analysis and the coding guidelines so as to allow all organization to leverage the flexibility granted by the GPSV [3]. In addition, the GPSV acknowledges that software components from many sources may be used to create the application; ECLAIR provides facilities that allow: (1) controlling the use of interfaces among components (see the discussion on `B.INDEPENDENCE` in Section 5.5); (2) checking which parts of the software that is not developed in-house (third-party libraries, off-the-shelf software, contract software, shareware) are exercised by the application through the ECLAIR service `B.SCOUT`; (3) working effectively on highly-configurable software, ensuring that the correct version is analyzed ruling out all possible tool configuration issues.

Table 4: Summary of ECLAIR coverage of GPSV's requirements: Section 5

Requirement		ECLAIR
5.1	Software Life Cycle Activities	✓[a]
5.2.1	Quality Planning	
5.2.2	Requirements	✓[b]
5.2.3	Design	
5.2.4	Construction or coding	✓[b,c,d,e]
5.2.5	Testing by the software developer	✓[b]
5.2.6	User Site Testing	
5.2.7	Maintenance and software changes	✓[f]

[a] ECLAIR and its documentation can be used in the creation of a software development life cycle based on requirements, traceability, coding standards, software metrics, and static analysis that prevents the introduction of defects before testing is reached.

[b] ECLAIR service `B.REQMAN` allows ensuring that all code is forward and backward traceable to documented requirements, including functional and capability requirements, requirements on inputs, outputs and interfaces, safety requirements and security requirements. `B.REQMAN` also allows tracing code to the tests and back. The integrated requirements management tool makes ECLAIR a cost-effective, complete solution for requirements-based development and testing.

[c] ECLAIR can be used to verify the compliance of source code to coding standards, such as the MISRA coding standards and BARR-C:2018. In turn, compliance with such standards ensures that the semantics of the source code is well defined and that certain classes of run-time errors cannot occur. ECLAIR can also be used to verify that the value of software metrics are inside prescribed ranges.

[d] The *ECLAIR Independence Checker* (service `B.INDEPENDENCE`) allows enforcing constraints about the software architecture and ensuring the interface of software items are respected. For software items executing on the same processor, `B.INDEPENDENCE` can ensure that segregation is effective at the source code level.

[e] When software is constructed by assembling together previously coded software components (e.g., from code libraries), the ECLAIR service `B.SCOUT` allows for the precise identification of those parts of the existing code are actually used. This tipically enables significant savings, as only the parts that are actually used need to be validated.

[f] ECLAIR can be integrated in automated regression testing systems in order to ensure that whenever a change is made, no new non-compliances to coding standards are introduced or source code metrics are exceeded as a result of maintenance updates.

5.4 HIS and Other Source Code Metrics

Source code metrics are recognized by many software process standards (and from MISRA) as providing an objective foundation to efficient project and quality management. One well known set of metrics has been defined by HIS (Herstellerinitiative Software, an interest group set up by Audi, BMW, Daimler, Porsche and Volkswagen).

The *HIS source code metrics* [5], while well established, include some metrics that are obsolete and miss others that are required or recommended by software process standards, such as those that allow estimating function coupling. For this reason, ECLAIR supplements HIS source code metrics with numerous other metrics that allow software quality to be assessed in terms of complexity, testability, readability, maintainability and so forth. Keeping track of these metrics also provides an effective and objective method to assess the quality of the software development process. The full set of metrics is available in all ECLAIR packages.

5.5 ECLAIR Support for *Segregation* in IEC 62304

In IEC 62304 parlance, a *medical device* can be composed of different subsystems, some of which are *software systems*. In turn a *software system* is composed of one or more *software items*, and each *software item* is composed of one or more *software units* or decomposable *software items*. *Software units* are not further decomposed for the purposes of testing or software configuration management.

IEC 62304, in Section "Software safety classification," broadly refers to the concept of "segregation" [9, 11, 4.3]:

> d) When a SOFTWARE SYSTEM is decomposed into SOFTWARE ITEMS, and when a SOFTWARE ITEM is decomposed into further SOFTWARE ITEMS, such SOFTWARE ITEMS shall inherit the software safety classification of the original SOFTWARE ITEM (or SOFTWARE SYSTEM) unless the MANUFACTURER documents a rationale for classification into a different software safety class [...] Such a rationale shall explain how the new SOFTWARE ITEMS are segregated so that they may be classified separately.

This concept is further elaborated in [9, 11, B.4.3]:

> The software ARCHITECTURE should promote segregation of software items that are required for safe operation and should describe the methods used to ensure effective segregation of those SOFTWARE ITEMS. Segregation is not restricted to physical (processor or memory partition) separation but includes any mechanism that prevents one SOFTWARE ITEM from negatively affecting another. The adequacy of a segregation is determined based on the RISKS involved and the rationale which is required to be documented.

The *ECLAIR Independence Checker* (service `B.INDEPENDENCE`) allows the formal specification and systematic checking of software architectural constraints, e.g., to enforce constraints about layering and to prevent bypassing of software interfaces. `B.INDEPENDENCE` is instrumental in proving independence among different software components.

6 ECLAIR Qualification and Validation in Compliance with IEC 62304, FDA QSR/QMS and the GPSV

IEC 62304 does not contain specific requirements on software tools and on their qualification. However:

- IEC 62304 Annex C.4.6, when presenting the coverage of PEMS requirements in IEC 60601-1:2005 + IEC 60601-1:2005 /AMD1:2012, relates, in Table C.3, suggests that clause 14.6.2 ("Risk control") of IEC 60601, where it says "Suitably validated tools and procedures shall be selected

and identified [...]" is covered by IEC 62304 clause 5.1.4 ("Software development standards, methods and tools planning");

- IEC 62304 Annex C.1 says that "[...] IEC 61508-3 [...] can be looked to as a source of methods, tools and techniques that can be used to implement the requirements in IEC 62304."

As IEC 62304 does not define how suitable validation is achieved, but refers to IEC 61508 with respect to tools, the approach defined in IEC 61508 Part 3 can be followed [10, Clause 7.4.4].

Section 6 of the GPSV, particularly Section 6.3 (*Validation of off-the-shelf software and automated equipment*), stresses the importance, for the device manufacturer, of ensuring the adequacy of software and tools acquired from third parties. This concerns, in particular, the software development tools, such as compilers and linkers as well as software verification tools like ECLAIR.

The GPSV gives the device manufacturer great latitude on how to conduct tool qualification. The ECLAIR functional safety ecosystem provides solutions that allow covering every concrete case with just the appropriate budget and effort: see Section 8 below for more details.

6.1 FDA QSR/QMS Requirements on Software Used in Production and the Quality System

While IEC 62304 and the GPSV do not define a specific tool qualification scheme, United States regulatory requirements impose obligations on manufacturers concerning the validation of computer software used in production or in the quality system.

In particular, 21 CFR 820.70(i) [6] requires that:

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

Therefore, when a software verification tool such as ECLAIR is used as part of the manufacturer's development process, quality system, release workflow, or regulatory evidence generation process, the manufacturer is responsible for validating the tool for its intended use.

The ECLAIR functional safety ecosystem and associated documentation facilitate such validation by providing:

- documented specifications of ECLAIR functionality and services;
- extensive internal validation activities and regression test suites;
- reproducible and deterministic analysis behavior;
- traceable configuration mechanisms suitable for controlled environments;
- automation support for integration into CI/CD pipelines;
- optional qualification kits and services providing structured evidence and validation artifacts.

This allows the device manufacturer to define a validation protocol appropriate to the intended use of ECLAIR (e.g., coding standard enforcement, architectural constraint checking, metric monitoring, security-focused static analysis), execute the validation activities, and retain objective evidence within the quality management system.

In this way, ECLAIR supports not only compliance with IEC 62304 and IEC 81001–5–1 life cycle activities, but also the regulatory requirement to validate software tools used within the production and quality system environment.

Role of ECLAIR in Compiler Qualification

ECLAIR is also instrumental in achieving *compiler qualification* by validation. This is a service offered in cooperation with our partner *Solid Sands b.v.* as part of their *Compiler Qualification Service*. In case the *SuperTest* compiler test and validation suite reveals defects in the compiler, compliance with functional safety and cybersecurity standards requires appropriate mitigations to be defined and be systematically enforced. Common mitigations include:

1. avoidance of the use of certain language constructs in certain contexts;
2. use of a third party tool to supplement the diagnostic messages not provided by the compiler (e.g., when exceeding translation limits or violating language constraints);
3. avoidance of the use of certain compiler option combinations.

ECLAIR supports all three kinds of mitigations:

1. compliance with the MISRA guidelines excludes the use of several language constructs in certain contexts, and the MISRA guidelines have been designed taking into account the fact that some language constructs are more likely to expose compiler defects than others;
2. ECLAIR checkers for MISRA C:2025 Rule 1.1 and for MISRA C++:2023 Rule 4.1.1 provide suitable diagnostic message for all the violations of the applicable language standard, including the exceeding of translation limits;
3. due to the way ECLAIR works (intercepting all calls to the compiler, linker, assembler and librarian), checks can be defined to ensure the unwanted compiler option combinations are not used.

In addition, ECLAIR's *CerTran* extension automates the configuration of *SuperTest* for compiler qualification by scanning the application build process and creating the exact test configuration files needed to cover all use cases. This can be completely incorporated into a *Continuous Integration system*. Not only does this save a considerable amount of time, but it also avoids configuration errors that can easily occur when scanning the build process manually.

7 The European Medical Device Regulation

The European Medical Device Regulation (MDR) went into force on 26 May 2021. The MDR stipulates the regulations for "placing on the market, making available on the market or putting into service of medical devices for human use and accessories for such devices." It also regulates "devices used in the context of a commercial activity to provide a diagnostic or therapeutic service to persons," even when the devices are not themselves placed on the market. Even devices that are not intended for medical use, such as "lasers and intense pulsed light equipment for skin resurfacing, tattoo or hair removal or other skin treatment" are covered by the MDR.

Until 2021, medical device manufacturers could choose between the *EU Conformité Européenne* (CE) mark or *US Food and Drug Administration* (FDA) approval. The CE mark was easier to obtain than FDA approval. However, FDA approval meant that the device was approved for use worldwide, whereas the CE mark was limited to the European Union.

Since 26 May 2021 manufacturers have to comply with the MDR if they want to access the EU market. There is an implementation period, but from 27 May 2024 no medical device may be placed on the EU market unless it conforms to the MDR and has a valid EU certificate of conformity.

The MDR is considerably more comprehensive than FDA market authorization. Among many other things, the MDR details the obligations of the involved economic operators, the marking process requirements for identification and traceability of devices, and the post-market surveillance of marketed devices. It also establishes penalties for non-compliance.

The MDR pays particular attention to software. In some circumstances, software is itself considered to be a medical device. The regulations clarify this: "when specifically intended by the manufacturer to be used for one or more of the medical purposes set out in the definition of a medical device, [software in its own right] qualifies as a medical device."

There are two clauses in the MDR that go beyond the general principle already present in directives 90/385/EEC and 93/42/EEC, namely that all safety practices employed must, at any time, comply with the *generally acknowledged state-of-the-art standards*.

MDR Clause 17.2:

- This clause states that, for any device that incorporates software, or for software that is a device in itself, the software must be developed and manufactured in accordance with the state-of-the-art standards. These standards must take into account the principles of development life cycle and risk management, including information security, verification and validation. In this context, IEC 81001–5–1 can be used as part of the state-of-the-art evidence for security life cycle activities for health software.

MDR Annex ll, Clause 6:

- This gives more detail on software verification and validation. In particular, it prescribes the provision of information concerning all aspects of software design, development process and evidence of verification and validation addressing all hardware configurations and operating systems.

Summarizing, for medical devices, at least in the EU, there is more than just IEC 62304, and there is more to just obtaining approval or a (revised) CE mark: when things go wrong, the consequences of not following the generally acknowledged state-of-the-art standards can be catastrophic. For this reason the reader working on medical devices is advised to consider compliance with functional-safety standards more in line with the current state-of-the-art, in addition to IEC 62304. For details see, e.g., ECLAIR Coverage of IEC 61508, ECLAIR Coverage of EN 50128, and ECLAIR Coverage of ISO 26262.

8 ECLAIR Qualification in Compliance with IEC 62304, IEC 61508, 21 CFR 820.70(i) and Other Standards

The ECLAIR functionality described above is qualifiable in compliance with all major functional safety standards: IEC 62304:2006 + Amd 1:2015; IEC 61508:2010; ISO 26262:2018; ISO 25119:2018 + Amd 1:2020; ISO 19014-4:2020; EN 50128:2011 + A2:2020; and EN 50657:2017 + A1:2023. TÜV SÜD audited BUGSENG software development and quality assurance processes for ECLAIR, as well as the internal validation activities performed by BUGSENG on each ECLAIR release. At the end of its assessment, TÜV SÜD awarded BUGSENG the "Software Tool for Safety Related Development" Certificate no. Z10 116151 0001 Rev. 01, attesting that the ECLAIR Software Verification Platform is suitable to be used in safety-related development projects according to: IEC 62304 for any software safety class; IEC 61508:2010 for any SIL (*Safety Integrity Level*); ISO 26262:2018 for any ASIL (*Automotive Safety Integrity Level*); ISO 25119:2018 for any SRL (*Software Requirement Level*); ISO 19014-4:2020 for any MPLr (*Machine Performance Level required*); EN 50128:2011 for any SIL; EN 50657:2017 for any SIL. The ECLAIR FuSa Pack provides all what is necessary to take advantage of ECLAIR certification by TÜV SÜD. In addition, the ECLAIR Qualification Kits for IEC 62304 provides further help to safety teams in charge of qualifying ECLAIR for use in safety-related projects where the dependence on the tool operational environment has to be taken into account: the kits contain documents, test suites, procedures and automation facilities that can be used by the customer to independently obtain all the required confidence-building evidence. BUGSENG also offers the ECLAIR Qualification Service, whereby qualified BUGSENG personnel undertakes almost all the qualification effort.

9 The Bigger Picture

ECLAIR is very flexible and highly configurable: it supports all kinds of software development workflows and environments.

ECLAIR is fit for use in mission- and safety-critical software projects: it has been designed from the outset to exclude configuration errors that would undermine the significance of the obtained results.

ECLAIR is developed in a rigorous way and carefully checked with extensive internal test suites (tens of thousands of test cases) and industry-standard validation suites.

ECLAIR is based on solid scientific research results and on the best practices of software development.

ECLAIR's unique features and BUGSENG's strong commitment to the customer, allow for a smooth transition to ECLAIR from any other tool.

BUGSENG's quality system has been certified by TÜV Italia (TÜV SÜD Group) to comply with the requirements of UNI EN ISO 9001:2015 for the "Design, development, maintenance and support of tools for software verification and validation" (IAF 33).

BUGSENG is an Arm's Functional Safety Partner, and is thus recognized as a partner who can reliably support their customers with industry leading functional safety products and services.

For More Information

BUGSENG srl
Via Fiorentina 214/C
I-56121 Pisa, Italy
Email: info@bugseng.com
Web: http://bugseng.com

bugseng

**no shortcuts,
no compromises,
no excuses:
software verification done right**

References

[1] R. Bagnara, A. Bagnara, and P. M. Hill. Formal verification of software architectural constraints. In DESIGN&ELEKTRONIK, editor, *embedded world Conference 2023 — Proceedings*, pages 271–279, Nuremberg, Germany, 2023. WEKA FACHMEDIEN, Richard-Reitzner-Allee 2, 85540 Haar, Germany.

[2] R. Bagnara, M. Barr, and P. M. Hill. BARR-C:2018 and MISRA C:2012 (with Amendment 2): Synergy between the two most widely used C coding standards. In DESIGN&ELEKTRONIK, editor, *embedded world Conference 2021 DIGITAL — Proceedings*, pages 378–391, Nuremberg, Germany, 2021. WEKA FACHMEDIEN, Richard-Reitzner-Allee 2, 85540 Haar, Germany.

[3] R. Bagnara, S. Stabellini, N. Vetrini, A. Bagnara, S. Ballarin, P. M. Hill, and F. Serafini. Bringing existing code into MISRA compliance: Challenges and solutions. In DESIGN&ELEKTRONIK, editor, *embedded world Conference 2024 — Proceedings*, pages 327–338, Nuremberg, Germany, 2024. WEKA FACHMEDIEN, Richard-Reitzner-Allee 2, 85540 Haar, Germany.

[4] M. Barr. *BARR-C:2018 — Embedded C Coding Standard*. Barr Group, www.barrgroup.com, 2018.

[5] H. Kuder et al. HIS source code metrics. Technical Report HIS-SC-Metriken.1.3.1-e, Herstellerinitiative Software, April 2008. Version 1.3.1.

[6] U.S. Department Of Health, Food Human Services, and Drug Administration. 21 CFR Part 820 — Quality System Regulation. https://www.ecfr.gov/current/title-21/chapter-I/subchapter-H/part-820, February 2026. Electronic Code of Federal Regulations, accessed February 2026.

[7] U.S. Department Of Health, Food Human Services, Center for Devices Drug Administration, Center for Biologics Evaluation Radiological Health, and Research. *General Principles of Software Validation; Final Guidance for Industry and FDA Staff*. CDRH, CBER, January 2002.

[8] U.S. Department Of Health, Food Human Services, Center for Devices Drug Administration, Center for Biologics Evaluation Radiological Health, and Research. *Cybersecurity in Medical Devices: Quality System Considerations and Content of Premarket Submissions*. CDRH, CBER, February 2026.

[9] IEC. *IEC 62304:2006: Medical device software — Software life cycle processes*. IEC, Geneva, Switzerland, May 2006.

[10] IEC. *IEC 61508-3:2010: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems — Part 3: Software Requirements*. IEC, Geneva, Switzerland, April 2010.

[11] IEC. *IEC 62304:2006/Amd 1:2015: Medical device software — Software life cycle processes — Amendment 1*. IEC, Geneva, Switzerland, June 2015.

[12] IEC. *IEC 81001-5-1:2021: Health software and health IT systems safety, effectiveness and security — Part 5-1: Security — Activities in the product life cycle*. IEC, Geneva, Switzerland, December 2021.

[13] IEC. IEC 81001-5-1:2021/ISH1:2025: Interpretation Sheet 1 — Health software and health IT systems safety, effectiveness and security — Part 5-1: Security — Activities in the product life cycle. Interpretation sheet, December 2025.

[14] MISRA. *MISRA C++:2023 — Guidelines for the use of C++17 in critical systems*. The MISRA Consortium Limited, Norwich, Norfolk, NR3 1RU, UK, October 2023.

[15] MISRA. *MISRA C:2025 — Guidelines for the use of the C language in critical systems*. The MISRA Consortium Limited, Norwich, Norfolk, NR3 1RU, UK, March 2025.

[16] Motor Industry Software Reliability Association. *MISRA C++:2008 — Guidelines for the use of the C++ language in critical systems*. MIRA Limited, Nuneaton, Warwickshire CV10 0TU, UK, June 2008.