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

Coverage of IEC 62304

1 Introduction to IEC 62304:2006

IEC 62304:2006, "Medical device software — Software life cycle processes," is an international standard issued by IEC [5]. 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 [7].[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*.

1.1 Role of ECLAIR in Ensuring Compliance with IEC 62304:2006

The ECLAIR Software Verification Platform can be used to comply with several of the requirements of IEC 62304 [7]. In addition, the ECLAIR Fusa Pack greatly simplifies obtaining all the confidence-building evidence that is required to make a solid argument justifying the use of ECLAIR in safety-related projects.

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

[2] Emphasized expressions are given a precise meaning in [7].

2 ECLAIR Coverage of IEC 62304 Requirements

The following table 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. 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. Note that, in the sequel, every reference to MISRA C:2012 should be interpreted as referring to [8] as amended by [9, 10, 11], whereas MISRA C++ is [12]. 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:2012 Directive 3.1" corresponds to the ECLAIR service `MC3R1.D3.1` 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.PROJORG` 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.

2.1 MISRA C:2012

MISRA C:2012 Revision 1 [8], with Amendments 2 [9] and 3 [10], and Technical Corrigendum 2 [11], 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:2012, which allows coding MISRA-compliant applications in subsets of C90, C99, C11 and C18, is supported by the ECLAIR package called "MC3".

2.2 MISRA C++:2008

MISRA C++:2008 [12] is the software development C++ subset developed by MISRA for the motor industry, which is now a de facto standard for safety-, life-, and mission-critical embedded applications also in many other industries. A new set of guidelines for C++17 is currently under development: these guidelines have adapted many of the existing guidelines in MISRA C++:2008, MISRA C:2012 and AUTOSAR as well as adding many new guidelines applicable to C++17. MISRA C++:2008 is supported by the ECLAIR package called "MP1".

2.3 BARR-C:2018

The *Barr Group's Embedded C Coding Standard*, BARR-C:2018 [3], 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".

2.4 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

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] ECLAIR service `B.PROJORG` 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.PROJORG` 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.

been defined by HIS (Herstellerinitiative Software, an interest group set up by Audi, BMW, Daimler, Porsche and Volkswagen).

The *HIS source code metrics* [4], 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.

2.5 ECLAIR Support for *Segregation* in IEC 62304

In IEC 62304 parlance, a *medical device* can be composed of difference 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" [5, 7, 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 [5, 7, 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.

ECLAIR service `B.PROJORG` 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.PROJORG` is instrumental in proving independence among different software components.

3 ECLAIR Qualification and Compliance with of IEC 62304

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 [6, Clause 7.4.4].

4 ECLAIR Qualification in the Medical Sector Beyond IEC 62304

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 recently, 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 devices that incorporate software, or for software that are devices in themselves, 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.

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.

5 ECLAIR Qualification in Compliance with IEC 62304, IEC 61508 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; EN 50128:2011 + A2:2020; and ISO 25119:2018 + Amd 1:2020. 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. 00, 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*); EN 50128:2011 for any SIL. ISO 25119:2018 for any SRL (*Software Requirement Level*).

6 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 Marco dell'Arpa 8/B
I-43121 Parma, Italy
Email: info@bugseng.com
Web: http://bugseng.com
Tel.: +39 0521 461640

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

[4] 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.

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

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

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

[8] MISRA. *MISRA C:2012 — Guidelines for the use of the C language critical systems*. HORIBA MIRA Limited, Nuneaton, Warwickshire CV10 0TU, UK, February 2019. Third edition, first revision.

[9] MISRA. *MISRA C:2012 Amendment 2 — Updates for ISO/IEC 9899:2011 Core functionality*. HORIBA MIRA Limited, Nuneaton, Warwickshire CV10 0TU, UK, February 2020.

[10] MISRA. *MISRA C:2012 Amendment 3 — Updates for ISO/IEC 9899:2011/2018 Phase 2 — New C11/C18 features*. The MISRA Consortium Limited, Norwich, Norfolk NR3 1RU, UK, October 2022.

[11] MISRA. *MISRA C:2012 Technical Corrigendum 2 — Technical clarification of MISRA C:2012*. The MISRA Consortium Limited, Norwich, Norfolk NR3 1RU, UK, March 2022.

[12] 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.