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

Coverage of ISO 25119

1 Introduction to ISO 25119:2018

ISO 25119:2018, "Tractors and machinery for agriculture and forestry — Safety-related parts of control systems," is a series of international functional-safety standards applicable to the design and development of all safety-related parts of control systems (SRP/CS) on tractors used in agriculture and forestry as well as on self-propelled ride-on machines and mounted, semi-mounted and trailed machines used in agriculture. The standard can also be applied to mobile municipal equipment, such as street-sweeping machines. It is a sector-specific implementation of the IEC 61508 series of standards and its first edition was published in 2010. The second edition, published in 2018, completely supersedes the previous version and contains significant technical revisions. ISO 25119 Part 2 has been subsequently revised in 2019 [7].

ISO 25119 approach to risk management is based on the determination of the *Agricultural Performance Level* (AgPL) for each safety-related part of control systems; the AgPL specifies the ability of such parts to perform a safety-related function under foreseeable conditions [5] There are six AgPL: *QM*, *a*, *b*, *c*, *d*, and *e*. The AgPLs *a*–*e* correspond to the *Performance Levels* (PL) *a*–*e* of ISO 13849-1:2015 "Safety of machinery — Safety-related parts of control systems." Thus, AgPL *e* corresponds to the most critical safety-related functions, i.e., those that have potential for severely life-threatening or fatal injury in the event of a malfunction and that, consequently, require a *Probability of Dangerous Failure per Hour* (PFHd) less than 10^{-7}.

As software is concerned, ISO 25119 defines how the *Software Requirement Level* (SRL) is derived from the AgPL [7, Section 7.3.5]. The SRL is categorized into four groups: B, 1, 2, and 3, which correspond to increasingly strict safety goals.

1.1 Role of ECLAIR in Ensuring Compliance with ISO 25119:2018

The ECLAIR Software Verification Platform can be used to comply with several of the techniques and measures required by ISO 25119:2018 Part 3 "Series development, hardware and software" [6]. 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.

2 ECLAIR Coverage of ISO 25119:2018 Techniques and Measures

Part 3 of ISO 25119:2018 specifies the requirements for product development at the software level [6]. It features several tables defining techniques and measures that must be considered in order to comply with the standard. The different techniques and measures listed in each table contribute to the level of confidence in achieving compliance with the corresponding requirement. Techniques and measures are listed in each table either as *consecutive entries*, numbered with 1, 2, 3, ... in the leftmost table column, or as *alternative entries*, labeled with 1a, 1b, 1c, ... in the same column.

The degree of recommendation to use each technique or measure depends on the SRL, and is symbolically encoded as follows:

+ indicates that the technique or measure shall be used for the SRL, unless there is reason not to, in which case that reason shall be documented during the planning phase;

o indicates that there is no recommendation for or against the use of technique or measure for the SRL;

x indicates that the technique or measure is not suitable for meeting the SRL.

For consecutive entries, all techniques/measures listed with **+**, in accordance with the SRL, apply. For alternative entries, only one of the techniques/measures shall be applied in accordance with the SRL.

The following tables have been obtained by extending the corresponding tables in ISO 25119:2018 Part 3 with a column indicating 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.

Table 1 — Software safety requirements specification

Technique/Measure		SRL				ECLAIR
		B	1	2	3	
1	Requirements specification in natural language	+	+	+	+	✓[a]
2a	Informal methods	+	+	x	x	–
2b	Semi-formal methods	+	+	+	+	–
2c	Formal methods	+	+	+	+	–
3	Computer-aided specification tools	o	o	+	+	✓[a,b]
4a	Inspection of software safety requirements	+	+	+	+	✓[a]
4b	Walk-through of software safety requirements	+	+	x	x	✓[a]

[a] ECLAIR service `B.REQMAN` allows ensuring that all code is forward and backward traceable to documented requirements, including safety 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] 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.

Table 2 — Software architecture design

Technique/Measure		SRL				ECLAIR
		B	1	2	3	
1a	Informal methods	+	+	x	x	–
1b	Semi-formal methods	+	+	+	+	–
1c	Formal methods	+	+	+	+	✓[a]
2	Computer-aided specification tools	o	o	+	+	✓[a,b]
3a	Inspection of software architecture	+	+	+	+	✓[a]
3b	Walk-through of software architecture	+	+	x	x	✓[a]

[a] 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] ECLAIR service `B.REQMAN` allows ensuring that all code is forward and backward traceable to documented requirements, including safety 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.

2.1 MISRA C:2025

MISRA C:2025 [9] 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".

2.2 MISRA C++:2023

MISRA C++:2023 [8] 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 [10], 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".

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".

Table 3 — Software design and development
Support tools and programming language
Section 1

	Technique/Measure	SRL				ECLAIR
		B	1	2	3	
1	**Tools and programming language**					
1.1	Suitable programming language	+	+	+	+	✓[a]
1.2	Strongly typed programming language	o	+	+	+	✓[b]
1.3	Use of language subsets	o	+	+	+	✓[c]
1.4	Tools and translators: increased confidence from use	o	+	+	+	✓[d]
1.5	Use of trusted/verified software components (if available)	o	o	+	+	✓[e]

[a] C and C++ have a long history of application in the development of embedded system. The MISRA subsets enforceable by ECLAIR are widely recognized in all industry sectors as suitable to the development of safety-critical systems.

[b] MISRA C/C++ enforce strong typing on the respective languages. E.g., for MISRA C:2025, Rules 10.1–10.8, 11.1–11.9 and 14.4; for MISRA C++:2023, Rules 7.0.1–7.0.6, 7.11.1 and 8.2.1–8.2.7.

[c] MISRA C/C++ and BARR-C:2018 define language subsets where the potential of committing possibly dangerous mistakes is reduced.

[d] ECLAIR is used across all industry sectors. It is also thoroughly validated by means of internal and third-party test suites. In addition, ECLAIR can verify the code adheres to a given version of C/C++, thereby ensuring that compiler qualification with respect to that version of C/C++ is relevant to the project. E.g., for MISRA C:2025, Rule 1.1 and Directive 1.2; for MISRA C++:2023, Rule 4.1.1.

[e] MISRA C/C++ guidelines (particularly those with *System* analysis scope) and their enforcement with ECLAIR allow using trusted/verified software components and libraries avoiding common pitfalls in the integration of components of different provenance.

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 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.

Table 3 — Software design and development
Support tools and programming language
Section 2

Technique/Measure		SRL				ECLAIR
		B	1	2	3	
2	**Design methods**					
2.1a	Informal methods	+	+	x	x	–
2.1b	Semi-formal methods	+	+	+	+	–
2.1c	Formal methods	+	+	+	+	✓[f]
2.2	Defensive programming	o	o	o	+	✓[g]
2.3	Structured programming	o	+	+	+	✓[h]
2.4	Modular approach					
2.4.1	Software component size limit	o	+	+	+	✓[i]
2.4.2	Software complexity control	o	o	o	+	✓[i]
2.4.3	Information hiding/encapsulation	o	o	+	+	✓[j]
2.4.4	One entry/one exit point in subroutines and functions	o	+	+	+	✓[k]
2.4.5	Fully defined interface	o	+	+	+	✓[l]
2.5	Library of trusted/verified software components	+	+	+	+	✓[e,m]
2.6	Computer-aided design	o	o	o	+	✓[n]

[e] MISRA C/C++ guidelines (particularly those with *System* analysis scope) and their enforcement with ECLAIR allow using trusted/verified software components and libraries avoiding common pitfalls in the integration of components of different provenance.

[f] The *ECLAIR Independence Checker* (service `B.INDEPENDENCE`) allows enforcing constraints about layering to prevent bypassing of software interfaces.

[g] The MISRA C/C++ guidelines promote the use of several defensive programming techniques. E.g., for MISRA C:2025, Directives 4.1, 4.7, 4.11 and 4.14, Rules 14.2, 15.7, 16.4, 17.4 and 17.7; for MISRA C++:2023, Rules 0.1.2, 9.4.1, 9.4.2, 9.5.1, 9.6.5 and 18.3.1.

[h] The MISRA C/C++ guidelines include limits on the use of non-structured control-flow constructs. E.g., for MISRA C:2025, Rules 14.3, 15.1–15.4, and 21.4; for MISRA C++:2023, Rules 0.0.2, 9.6.1–9.6.3 and 21.10.2. A threshold on metric `HIS.GOTO` allows limiting the use of `goto`.

[i] HIS [4] and other metrics related to the size and complexity of software components. ECLAIR allows associating thresholds to each metric.

[j] The MISRA C/C++ guidelines promote the use of information hiding and encapsulation. E.g., for MISRA C:2025, Directives 4.3 and 4.8 and Rules 8.7 and 8.9; for MISRA C++:2023, Rules 6.0.3, 6.5.1 and 6.7.2. In addition, the *ECLAIR Independence Checker* can be used to enforce strict encapsulation constraints.

[k] MISRA C:2025 Rule 15.5 and MISRA C++:2008 Rule 6-6-5 require subprograms to have a single entry and a single exit only. An upper threshold on metric `HIS.RETURN` allows for a more flexible approach.

[l] The MISRA C/C++ guidelines promote the full definition of interfaces. E.g., for MISRA C:2025, Rules 8.2 and 8.3 prescribe the use of prototype form and the use of consistent names and qualifiers for function declarations, Rule 17.3 forbids implicit declarations, Directive 4.14 requires data verification; for MISRA C++:2023, Rules 6.2.2, 6.9.2 and 13.3.3 prescribe the use of consistent name and types for entity declarations. BARR-C:2018 Rule 2.2.h recommends commenting modules and functions with explicit specification of pre-conditions and post-conditions with Doxygen; such comment blocks are automatically checked by ECLAIR for consistency.

[m] The *ECLAIR Scout* product (service `B.SCOUT`) allows for the precise identification of the component to be qualified. This is particularly challenging in the case of C++, where libraries make extensive use of templates and compile-time evaluation, so that much code has not a direct manifestation in the object code.

[n] Many ECLAIR services are in fact computer-aided design tools.

Table 3 — Software design and development
Support tools and programming language
Sections 3 and 4

Technique/Measure		SRL				ECLAIR
		B	1	2	3	
3	**Design and coding standard**					
3.1	Use of coding standard	o	+	+	+	✓[o]
3.2a	No dynamic variables of objects	o	o	o	+	✓[p]
3.2b	Online checking of the creation of dynamic variables	o	o	o	+	✓[q]
3.3	Limited use of interrupts	o	o	o	+	–
3.4	Defined use of pointers	o	o	o	+	✓[r]
3.5	Limited use of recursion	o	o	o	+	✓[s]
4	**Design and coding verification**					
4a	Inspection of software design and/or source code	+	+	+	+	✓[t]
4b	Walk-through of software design and/or source code	+	+	x	x	✓[t]

[o] The MISRA C/C++ and BARR-C:2018 coding standards define language subsets where the potential of committing possibly dangerous mistakes is reduced. Regarding coding style, more than half of the guidelines in BARR-C:2018 [3] concern coding style [2]. MISRA C:2025 Rules 7.3 and 16.5 and MISRA C++:2023, Rules 5.13.5, 5.13.6 and 6.0.1 are also stylistic.

[p] The MISRA C/C++ guidelines include prescriptions limiting the use of manual dynamic memory allocation. E.g., for MISRA C:2025, Directive 4.12 and Rules 18.7, 21.3, 22.1 and 22.2; for MISRA C++:2023, Rules 21.6.1–21.6.3.

[q] The MISRA C/C++ guidelines require systematic checking of error information returned by functions. Guidance is also provided on how to perform some of these checks. E.g., for MISRA C:2025, Directive 4.7, Rules 17.7, 22.8, 22.9, and 22.10; for MISRA C++:2023, Rules 0.1.2 and 18.3.1

[r] The MISRA C/C++ guidelines include rules restricting the use of pointers. E.g., for MISRA C:2025, Rules 8.13, 11.1–11.8, and 18.1–18.5; for MISRA C++:2023, Rules 8.2.3, 8.2.4, 8.2.6–8.2.8, 8.7.1, 8.7.2, 8.9.1 and 10.1.1. The specific ECLAIR services `B.PTRDECL` and `B.PTRUSE` allow fine control of pointers' use.

[s] MISRA C:2025 Rule 17.2 and MISRA C++:2023 Rule 8.2.0 forbid recursion. A threshold on metric `HIS.ap_cg_cycle` also allows ruling out recursion.

[t] Compliance to the MISRA C/C++ and the BARR-C:2018 guidelines greatly increases code readability and understandability, thereby facilitating verification activities by walk-through, pair-programming and inspection.

Table 4 — Software component testing
Section 1

Technique/Measure		SRL				ECLAIR
		B	1	2	3	
1	**Static analysis**					
1.1	Boundary value analysis	+	+	+	+	–
1.2	Checklists	o	o	o	o	–
1.3	Control flow analysis	o	o	+	+	✓[a]
1.4	Data flow analysis	o	o	+	+	✓[b]

[a] ECLAIR builds accurate control flow graphs to reason on (feasible and unfeasible) execution paths.
[b] ECLAIR performs a number of data flow analyses to reason about, e.g., pointers, values and dead stores.

3 ECLAIR Qualification in Compliance with ISO 25119 Part 3

The requirements of ISO 25119 on providing evidence for correct tool behavior are not substantially different from the ones of the base norm IEC 61508. Part 3 of ISO 25119:2018 specifically recommends the use of tools as follows [6]:

> Computer-aided (CAD) tools shall be used during the design of both hardware and software when available and justified by the complexity of the system. The correctness of such tools shall be demonstrated by specific testing, an extensive history of satisfactory use, or by independent verification of their output for the applicable safety-related system.

Part 1 of ISO 25119:2018 poses the following requirement for the use of any tool [5]:

> If tools are used for development, implementation or testing, their application shall be assessed or verified.

The ECLAIR functionality described above is qualifiable in compliance with ISO 25119. 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 ISO 25119:2018 for any SRL.

The ECLAIR Qualification Kits for ISO 25119 provide 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.

4 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] 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] ISO. *ISO 25119:2018: Tractors and machinery for agriculture and forestry — Safety-related parts of control systems — Part 1: General principles for design and development*. ISO, Geneva, Switzerland, October 2018.

[6] ISO. *ISO 25119:2018: Tractors and machinery for agriculture and forestry — Safety-related parts of control systems — Part 3: Series development, hardware and software*. ISO, Geneva, Switzerland, October 2018.

[7] ISO. *ISO 25119:2019: Tractors and machinery for agriculture and forestry — Safety-related parts of control systems — Part 2: Concept phase*. ISO, Geneva, Switzerland, August 2019.

[8] 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.

[9] 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.

[10] 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.