The development of high-quality software is a tough task: ECLAIR has been designed to help development and quality assurance teams achieve that quality.

MISRA-C++:2008 + HIS Metrics Package

The *MISRA-C++:2008 + HIS Metrics* package is a combination of two of the many applications that run on top of ECLAIR, a powerful platform for the automatic analysis, verification, testing and transformation of C and C++ programs (with an ongoing extension to Java and a planned extension to other languages). This particular package combines:

- a state-of-the-art, medium-weight static analyzer suitable for execution on the developer's desktop that almost completely automates the assessment of conformance with respect to MISRA-C++:2008;
- a precise and flexible implementation of the source code metrics defined by HIS.

1 Highlights

- Proper coverage of MISRA-C++:2008, not just a version of MISRA-C in disguise: the language and many rules are radically different and require completely different checkers.
- No time wasted in writing compiler personality files (often of questionable correctness).
- Automatic production of accurate, faithful and (optionally) tamper-proof compliance matrices.
- Easy-to-use web-based configuration interface with different configuration privileges.
- *Real-time* use from within most popular IDEs or *batch* use with reports stored in a database for later processing.
- Guideline violation and metric reports optionally available to the entire development team and management using web-based technology.
- Powerful mechanisms of differential reporting allow correlating changes in the code and the appearance/disappearance of violations (with possible interfaces to issue-tracking systems).
- No stress: free consultancy services for the initial configuration. This includes full assistance to help your company make the transition to the *MISRA-C++:2008 + HIS Metrics* package.

2 MISRA-C++:2008

MISRA-C++:2008 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 in many industries including aerospace, railway, medical, telecommunications and others.[1]

2.1 Coverage and Precision

ECLAIR's *MISRA-C++:2008 + HIS Metrics* package offers one of the most extensive, properly said MISRA-C++:2008 coverages available on the market, by providing support for around 87% of the guidelines.

Guidelines are enforced using very general and *accurate* checkers, which operate on the precise sequences of tokens and abstract syntax trees that are manipulated by the compiler.[2] Coupled with the fact that ECLAIR always checks each guideline in the appropriate context (at the token, declaration, translation unit, whole program or whole system levels), this makes sure that the checkers for decidable rules are *exact* (neither false positives nor false negatives). For undecidable rules, ECLAIR's *MISRA-C++:2008 + HIS Metrics* package provides a medium-weight solution to the tradeoff among computational complexity, number of false positives and number of false negatives. In any case, when false negatives are possible, they are always clearly and unambiguously delimited.

Coverage of the MISRA-C++:2008 guidelines is summarized in the following table:

Support level	**#**
Fully supported (without false negatives)	182
Partially supported (with possible false negatives)	16
Not yet supported	30
Total	228

2.2 Compliance Matrices

ECLAIR can be configured to automatically produce compliance matrices required to meet contractual obligations and industrial standards such as ISO 26262. The compliance matrix is obtained from the actual configuration, which, if properly done, will contain the reason for each deviation. Thus, carrying its rationale, any deviation goes straight from the configuration to the matrix.

In addition, thanks to ECLAIR's ability to intercept and fully understand the communication with the toolchain, the compliance matrix contains full details about the code and its analysis: which files have been compiled and/or analyzed (with full path and a cryptographic hash of their contents), the compiler/linker options, the full version of ECLAIR, ..., with even a cryptographic hash of the generated executables. All this allows the linking of the MCU's ROM actual content with the compliance matrix.

As an option, ECLAIR can be supplied with anti-tampering measures to produce a digitally-signed PDF compliance matrix whose integrity can be checked by third parties.

[1] 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.

[2] This is in sharp contrast with checkers that are based on pattern matching and imprecise parsing. If, on the one hand, they can deal with programs that do not compile, on the other hand they are plagued by a high number of false positives and, most importantly, false negatives, something that makes them unsuitable to safety-critical and mission-critical contexts. Generally speaking, beware of tools based on obsolete technology: liability of the manufacturer can only be waived if the defect could not be detected by the manufacturer using the *current state of science and technology at the time of production*.

3 HIS Source Code Metrics

The HIS[3] Software Test Working Group, recognizing the fact that software metrics provide an objective foundation to efficient project and quality management, specified a set of metrics to be used in the evaluation of software.[4] These metrics allow software quality to be assessed in terms of complexity, testability, readability, maintainability and so forth, and the quality of the software development process.

3.1 Coverage

ECLAIR's *MISRA-C++:2008 + HIS Metrics* package provides very precise and flexible coverage for all the HIS metrics with boundary limits apart from S_i, namely: COMF, PATH, GOTO, v(G), CALLING, CALLS, PARAM, STMT, LEVEL, RETURN, VOCF, NOMV, NOMVP and ap_cg_cycle. These measures may be incrementally reported, showing exactly where in the code the value was computed, or aggregated (e.g., maximized, summed, averaged) over a single function, translation unit, program, or the whole project. If a limiting value for a metric is provided, ECLAIR can report where this value is attained and also, if needed, each subsequent point in the code where a value that breaches the limit is computed. Moreover, the aggregate values may be themselves aggregated so that, for example, ECLAIR can report not only the cyclomatic complexity of each function, but also the maximum and average complexity for the translation unit and, if required, the maximum and average of each of these translation unit values over the whole project.

Support for HIS metrics S_{change}, S_{del} and S_{new} (and consequently S_i, which depends on those) is under development and will be shipped as a product update. These metrics will be automatically computed from the data produced by ECLAIR over two codebase revisions and the *diff* that separates these revisions. Such a diff can be directly obtained from common revision control systems (RCS) or, for those who do not have access to the development RCS, produced by a file comparison utility.

4 Proper Integration with the Toolchain

ECLAIR intercepts every invocation of the toolchain components (compilers, linker, assembler, archive manager) and it automatically extracts and interprets the options that the build system has passed to them. This allows for the seamless integration with any build system. Moreover, the user does not need to engage in error-prone activities such as: (a) specifying which files compose the application and where the right header files are located; (b) configuring the static analyzer so that the analysis parameters match the options given to the compilers (several options *do* affect the program semantics); (c) writing down predefined macros and the architectural parameters such as sizes, alignment constraints, address spaces and so forth. All this is automatic and supports build processes that involve the automatic generation of source files that depend on the configuration, without requiring the development and maintenance of a separate analysis procedure: with ECLAIR the existing build procedure can be used verbatim.

ECLAIR is available on most modern flavors of UNIX®, Linux, OS X® and Windows®, including Cygwin and MinGW, and can be used with just about any development environment. ECLAIR can leverage the availability of computing resources by supporting parallel and distributed program analyses. Most popular C/C++ compilers and cross compilers are supported, including CodeWarrior™, Cosmic Software, GCC, Green Hills®, IAR™, Intel®, Keil Software®, Microsoft®, MPLAB®, QNX™, Renesas Electronics, SOFTUNE™, TASKING®, Texas Instruments™, Wind River®, and clang/LLVM.

[3] HIS, Herstellerinitiative Software, is an interest group set up by Audi, BMW, Daimler, Porsche, and Volkswagen in order to join forces on methods of software design and quality assurance for microprocessor based control units.

[4] HIS source code metrics. Report HIS-SC-Metriken.1.3.1-e, Herstellerinitiative Software, April 2008. Version 1.3.1.

5 Web-Based User Interfaces

All the verification tasks supported by ECLAIR can be specified and refined incrementally by means of a very convenient web-based configuration interface. From any web browser this allows, for instance: finding coding rules using a powerful tag-based selection logic; activating and customizing coding rules, possibly restricting their use to only part of the project; selecting and customizing the kind of reports and compliance matrices to be generated; choosing to run the verification task immediately or save the task for later. Configurability of the checkers can be limited according to access privileges so that, e.g., only properly authenticated users can add certain kinds of deviations to the MISRA-C++:2008 guidelines. All deviations will in any case be reported into the compliance matrices.

Violation reports can be browsed using popular IDEs like Eclipse, Microsoft Visual Studio®, IAR Embedded Workbench®, Texas Instruments Code Composer Studio™, Keil µVision®, or any suitable editor. Violation reports can also be inspected using any web browser.

6 The Bigger Picture

ECLAIR is very flexible and highly configurable. It can support your software development workflow and environment, whatever they are. You can ask us to bend it to your precise needs or do that yourself.

ECLAIR is fit for use in mission- and safety-critical software projects: it has been designed from the outset so as to exclude false negatives unless the user's configuration explicitly asks for them.

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, such as ACE SuperTest and the Plum Hall Validation Suite for C.

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

ECLAIR is developed by a passionate team of programming language and software verification experts.[5] Do not hesitate to let us have your feedback: you may be surprised to discover just how much your suggestions matter to us.

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

Similar packages with MISRA-C:1998, MISRA-C:2004 and MISRA-C:2012 are also available!

For More Information

BUGSENG srl
Parco Area delle Scienze 53/A
I-43124 Parma, Italy
Tel: +39 0521 906 906
Fax: +39 0521 906 950
Email: info@bugseng.com
Web: http://bugseng.com

bugSeng

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

[5] Among them is the Italian representative within ISO JTC1/SC22/WG14, the international standardization working group for the programming language C.