Static Analysis

ASTRÉE


# Accelerate Finding All Runtime Errors & Data Races in C/C++ With Sound Static Analysis

QA SYSTEMS
The Software Quality Company

**Driving embedded software quality**

ASTRÉE

# Sound static analysis of C/C++

## Key Benefits

Seamless integration with QA-MISRA®

Scalable analysis from individual modules to 10 million lines of code

Sound analysis based on a mathematically rigorous formal method

Sound data and control coupling analysis / software component interference analysis

Interactive visualization of call graphs & classes

Automatic OS-aware analysis (ARINC-653, OSEK, and AUTOSAR)

Taint analysis to detect SPECTRE vulnerabilities, and user-defined cybersecurity analyses

Full traceability of reported code issues

Interactive traceable results & classification

Configurable report file generation

Tracking and visualization of project progress and analysis revisions

Stand-alone tool with open interfaces and open file formats

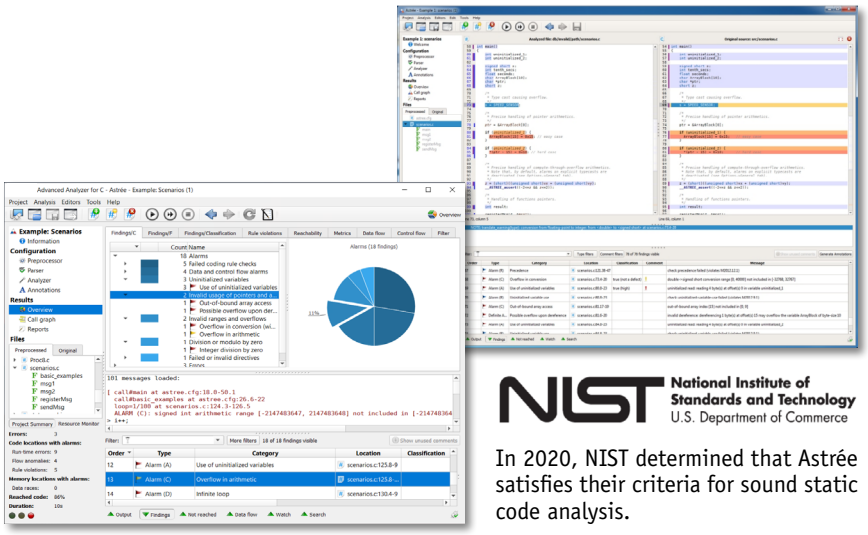
Automatic tool qualification for all major software safety standards

## The challenge

Runtime errors and data races can provoke erroneous program behaviour and may even cause the software to crash. Software testing can be used to detect errors, but not to prove their absence, since usually no complete test coverage is possible. Static analysis based on Abstract Interpretation can be used to prove the absence of runtime errors and data races. A small number of false alarms is important to enable an efficient verification process.

## The safe static analysis solution

Astrée is a parametric static analyzer designed to prove the absence of runtime errors and data races.
Astrée is sound, i.e., if no errors are signalled, this means there are no errors from the class of errors under investigation, the absence of errors has been proven. It reports program defects caused by unspecified and undefined behaviours according to the C and C++ language standards, program defects caused by invalid concurrent behaviour, and computes program properties relevant for functional safety.



NIST National Institute of Standards and Technology U.S. Department of Commerce

In 2020, NIST determined that Astrée satisfies their criteria for sound static code analysis.

## All the standards for C and C++ in one tool

With Astrée there are no hidden extras, coding language variants, or compliance module add-ons. It provides a single solution to automatically prove the absence of runtime errors, data races and further critical program defects in C/C++ code for the following language versions:

- ISO/IEC9899:1990 (C90)
- ISO/IEC9899:1999 (C99)
- ISO/IEC9899:2011 (C11)
- ISO/IEC9899:2018 (C18)
- ISO/IEC 14882:2003 (C++03)
- ISO/IEC 14882:2011 (C++11)
- ISO/IEC 14882:2014 (C++14)
- ISO/IEC 14882:2017 (C++17)

ASTRÉE

## Error classes reported


- Integer/floating-point division by zero.
- Out-of-bounds array indexing.
- Erroneous pointer manipulation and dereferencing (null, uninitialized, and dangling pointers).
- Data races (read/write or write/write concurrent accesses by two threads to the same memory location without proper mutex locking).
- Inconsistent locking (lock/unlock problems).
- Invalid calls to operating system services (e.g. OSEK calls to `TerminateTask()` on a task with unreleased resources).
- Integer and floating-point arithmetic over-flows.
- Read accesses to uninitialized variables.
- Code Astrée can prove to be unreachable under all circumstances (note that this is not necessarily all unreachable code).
- Violations of optional user-defined assertions to prove additional runtime properties. These static assertions can be formulated with arbitrary side-effect free C expressions. When Astrée does not report an assertion failure alarm, the correctness of the asserted expression has been formally proven.
- Violations of coding rules and code metric thresholds enhance QA-MISRA compliance checks.
- Non-terminating loops.

Astrée is sound for floating-point computations and handles them precisely and safely. It takes all potential rounding errors into account.

## Sound data & control flow coupling

Astrée computes data and control flow reports containing a detailed listing of accesses to global and static variables sorted by functions or variables, and caller/callee relationships between functions.

The analyzer can also report each potentially shared variable, the list of asynchronous tasks accessing it, and the types of the accesses (read, write, read/write).

## Configurable analysis & reports

Flexible configuration for analysis projects with mixed C & C++, autogenerated code or manually written code. These configurable rulesets can be selected in the GUI or stored in configuration files for command line batch execution.

Astrée generates summary results and configurable reports to explore in the GUI, and in multiple open standard formats for off-line compliance evidence.

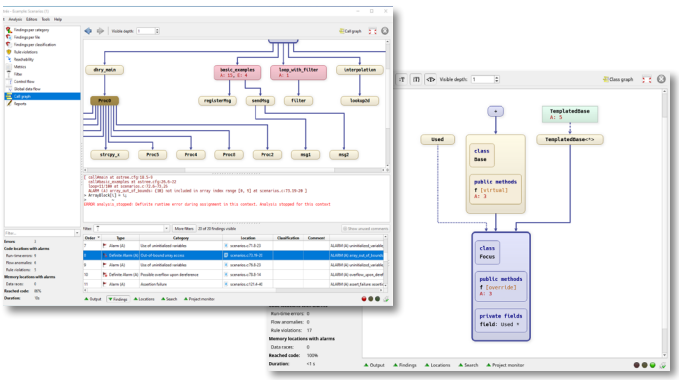
## Zero false negatives & false positives

Astrée produces zero false negatives and zero false positives on syntactic checks, and zero false negatives and very low false positives on semantical checks. No false negatives means the analysis scope did not miss any defects. Lower false positives mean less wasted effort on assessing and discarding as 'false' a reported violation (alarm).

## Visualizations aid understanding

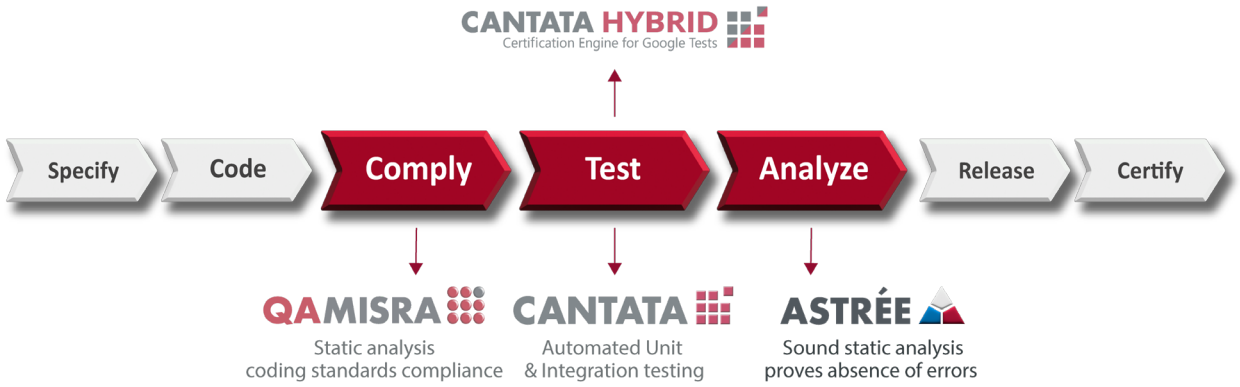
The interactive visualisations of call graphs (including pointers) are linked to code and results views. They show complete and partial call graphs for any analysed program, including thread specific calls.

A filterable view in the GUI presents all classes and class relationships for the selected code, including those defined inside the C++ standard namespace. A class graph visualizes a selected class together with its relations to others. Possible relations are inheritance, template instantiation/specialization, and usage as type of field members. For each individual class, detailed information on the respective field and method members can be inspected.



# Verification Centric Tools

QA Systems static analysis and software testing tools support verification in the linear flow of software development below. We recommend applying sequential approach to these verification stages with tools that are designed and targeted for each purpose.




**Comply** Use **QA-MISRA** for fast coding standard compliance at the developer's desktop first.

**Test** Use **Cantata** for automated dynamic execution of the standard compliant software.

Use **Cantata Hybrid** to generate certified Cantata test results from existing Google tests.

**Analyze** Use **Astrée** for proving absence of run-time errors on whole application.

NB: Astrée uses the same configuration as QA-MISRA, so the effort to apply it later to a QA-MISRA project is low.

## Platforms & Integrations

Astrée is available on Linux and Windows platforms. Pre-processing and parsing knowledge of the compilation environment is captured in project set-up and used in the analysis to support embedded software. In addition to the GUI client and full command line interfaces, Astrée is integrated with the following:

- Eclipse® (including integration with Cantata)
- ARM Keil $\mu$Vision IDE / Debugger®
- Jenkins®
- dSPACE TargetLink®
- MATLAB /SIMULINK®

## Start free trial

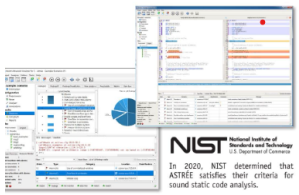
Take Astrée for a test drive with your own code and development environment.

Get in touch with our team at sales@qa-systems.com

## What to expect

- A bespoke demo kick-start and Q&A session
- Very simple installation
- Simple configuration for your compiler
- Unrestricted use, time-limited trial license

## Learn more




qa-systems.com/tools/astree

QA SYSTEMS
The Software Quality Company

**QA Systems Group** | **sales@qa-systems.com** | **www.qa-systems.com**
With offices in **Stuttgart, Germany** | Bath, UK | Milan, Italy |
Boston, United States | Da Nang, Vietnam | Bengaluru, India