QA·C++ STATIC ANALYZER

ENSURING QUALITY AND SECURITY AT THE APPLICATION LEVEL

ENTERPRISE-GRADE AUTOMATED SOURCE CODE ANALYSIS

QA·C++ detects and reports on dataflow problems, software defects, language implementation errors, inconsistencies, dangerous usage and coding standard violations quickly and efficiently. By adhering to the "early & often" philosophy, software defects are identified at creation resulting in simplified development lifecycle and reduced costs and cycle time. QA·C++ provides an efficient, robust, and fully automated environment to introduce and enforce coding standards. A multitude of reports on detected defects as well as metric calculations can be created, and personalized. QA·C++ provides the ability to monitor complexity and highlight cases that exceed defined thresholds, enabling the development of testable and maintainable code.

The QA·C++ analyzer is the core component of the PRQA source code analytics platform and provides the analysis and reporting capabilities to directly highlight violations of the ISO guidelines, and combines error detection and security best practice with full integration within the PRQA product suite.

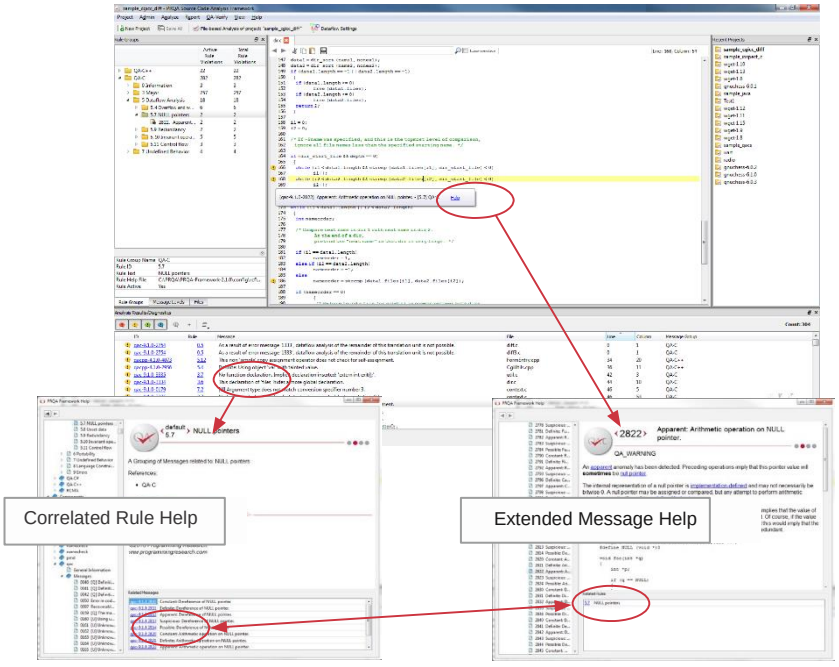
QA·C++ IDENTIFIES WHAT THE PROBLEM IS, EXPLAINS WHY IT'S A PROBLEM AND SHOWS HOW TO FIX IT

The QA·C++ static analyzer automatically performs in-depth analyses on your source code without executing programs. It checks your software for reliability, security, and conformance to ISO coding best practices and can be configured to run locally on either desktop or server. QA·C++ identifies issues which compilers and most developers miss. These include lesser-known issues explicitly stated in the ISO standards and language constructs that, while not classified as incorrect, may result in unpredictable behavior.

Unlike bug catchers or less sophisticated static analyzers, QA·C++ finds more issues while producing fewer false positives and negatives.

BENEFITS:

- Scale to millions of lines of code
- Improve the overall quality and security of any application
- Increase code portability and re-usability
- Continuously inspect source code for conformance to the coding standard of your choice
- Give your developers contextual feed-back that helps them correct and learn from mistakes
- Reduce bottlenecks caused by manual code review and slow analysis tools and methods
- Analyze your source code without executing programs



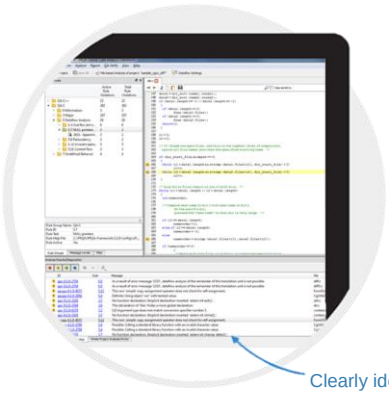
> *We found that 25% of the defects (...) would have been identified earlier by QA·C++ (during the coding phase). Our analysis concluded that it took us on average 2 man days longer to fix any defect discovered later in the process. The payback on QA·C++ was less than 18 months."*
>
> Robin Sayce-Jones, Senior Software Engineer, Trailer Systems at Haldex

KEY FEATURES

ADVANCED DEFECT PREVENTION

Using a proprietary, high-performance C++ language parser combined with a Deep Flow Dataflow analysis engine, QA·C++ is able to build an accurate model of the behavior of the software and track the value of variables in the code as they would be at run time. This sophisticated analysis approach maximizes code coverage while minimizing false positives and false negatives and allows QA·C++ to detect critical defects not reported by compilers or other tools and recognize issues caused by dangerous, overly complex and non-portable language usage.

Identify unpredictable behaviors others miss

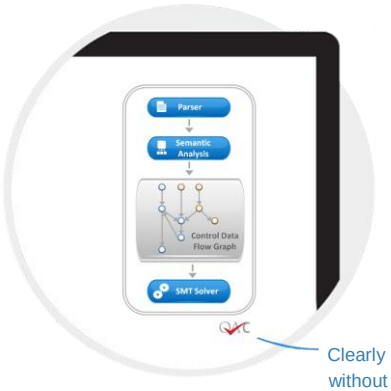


Clearly identify errors without executing code

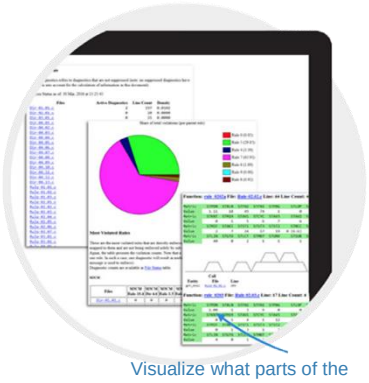
ACTIONABLE RESULTS

QA·C++ clearly identifies must-fix defects and includes a comprehensive knowledge base help system that provides detailed guidance with examples to support developers in fixing the issues found in the source code. Because developers get immediate and contextual feedback within their development environment, they can make the required changes as they are creating new code or reviewing existing code. In this way, developers build awareness of best practice approaches and can quickly form coding habits that are aligned with your organization's expectations.

QA·C++ identifies critical coding issues relating to control-flow, variable state, library usage and semantic modelling of your code. QA·C++ Dataflow analysis engine incorporates an advanced, industry-proven Satisfiability Modulo Theories (SMT) solver engine – a technology first for deep-flow static analysis products.



Clearly identify errors without executing code



Visualize what parts of the code need the most atteintion

MONITOR AND CONTINUALLY IMPROVE YOUR CODEBASE WITH CONFIGURABLE REPORTS

The compliance report helps you to visualize which areas of your codebase require the most attention to reach a higher-level compliance.

The code review report refocuses peer review on discussing design, optimization, and meeting requirements rather than costly manual investigation of code conformance and correctness.

The metrics data report provides you with an XML file that you can use as a source of quality metrics data for your own further examination.

The suppression report provides information on message diagnostics that have been suppressed during analysis.

ANALYSIS OF INDUSTRIAL-SCALE CODE

Automated static analysis using QA·C++ assists in identifying defects, vulnerabilities, and compliance issues early in the development cycle where they can be fixed faster and at lower cost. QA·C++ is fast, non-disruptive, easy-to-use, and scales to any size of development environment. As a result, organizations whose products need to perform securely and reliably in mission critical and safety critical environments trust in QA·C++ to help lower the risk of software failures, improve quality and reduce time-to-market.

EASY TO LEARN AND EASY TO USE

QA·C++'s powerful GUI delivers a contextual drill-down environment linked to a deep knowledge base. QA·C++ explains why problems it discovers need to be corrected and then provides guidance to help in fixing them.

ADVANCED DATAFLOW DETECTION

QA·C++ detects buffer overflows, division by zero, dead code, unreachable code and much more by linking the in-depth language analysis performed by QA·C++ with a state of the art inter-function cross translation unit dataflow Satisfiability Modulo Theories (SMT) solver. Wide scope of checking includes: inter-dependency between variables, pointer aliasing, bi-directional suspicious variable usage analysis and loop analysis (first, last and intermediate iteration analysis).

ADAPTABLE TO FIT EXISTING DEVELOPMENT ENVIRONMENTS

QA·C++ can be easily integrated into existing build systems and continuous integration environments to provide a means to enhance "early and often" testing with automated code analysis that helps to avoid errors that are expensive to fix late in the development cycle. This allows existing code review processes to be accelerated and refocused, thereby helping to increase overall productivity while also improving quality and security of the software. Additionally, QA·C++ can be configured for incremental analysis to ensure that only new changes are analyzed and feedback can be provided quickly.

ROBUST AND FLEXIBLE CODING STANDARD ENFORCEMENT

QA·C++ can be supplemented with optional modules that automate compliance checks for major coding standards and the generation of the reports and audit documentation required to demonstrate compliance. Custom coding standards can also be enforced by configuring the rules that should be applied to check for general language and project or domain specific issues. QA·C++ also allows messages to be suppressed at targeted source code locations and these suppressions can be audited to report deviations to enforcement of a standard for compliance purposes.

Available compliance modules for major standards include MISRA C++:2008, HIC++, JSF AV C++, CERT C++ and CWE C++.

KEY CHECKS

Avoid constructs in the C++ language that can reduce code reusability and lead to product failures, functional safety issues and vulnerabilities that attackers can exploit. Some of the risks that QA·C++ helps you avoid include:

- Undefined behavior
- ISO language constraint violations
- Overflow and wraparound (including division by zero)
- Uninitialized data
- Memory/pointer operation issues (including null pointer dereference)
- Control flow issues
- Type conversions
- Redundant code
- Shift operations
- Invariant operations
- Object/function declaration and definition issues
- Dangerous language use
- Non-portable language use
- Naming conventions for identifiers
- Best practice violations use of tainted data

TECHNICAL SPECIFICATIONS

GENERAL FEATURES
- Command line interface (CLI)
- Interactive GUI with message browser
- Online help & knowledge base
  - Usage & implementation contextual message
  - C++ language
  - Coding standard specific
- Summary & detailed reports
- IDE integrations
- Support for C++11 and C++14

CODE ANALYSIS FEATURES
- 1,500+ selectable messages
- C++ language-specific parsing engine
- Parses code of any size & complexity
- Handles common language extensions
- Cross module analysis (link time checking)
- Semantic error detection
- Inter-function and cross-TU
- Dataflow error detection
- Close name analysis

MESSAGE OUTPUT CONTROL
- Comment based suppression
- Baselining

METRICS
- Project based: 5
- File based: 16
- Function based: 30
- Class based: 13
- Warnings on metrics threshold values

RESULTS OUTPUT
- Configurable HTML reports
- Standard report types
  - Compliance
  - Code review
  - Suppression
  - Metric Data

CODING STANDARD ENFORCEMENT
- User configurable coding standards
- Add-on modules
  - MISRA C++: 2008
  - HIC++
  - JSF AV C++
  - CERT C++
  - CWE C++
- ISO C++ standard support
- Rule subsets for legacy code
- Best practice issues
- Naming convention checker
- Layout checker
- Defensive programming—defect avoidance
- Extensible rule base
- Customizable message text
- Deviation support

ISO C++ STANDARD SUPPORT
- Full checking of ISO C++ constraints
  - Undefined behavior
  - Unspecified behavior
  - Implementation defined behavior
- The tool will also identify language usage which is not compliant with the modern ISO C++ standards ISO/ IEC 14882:2014(E) and ISO/IEC 14882:2011(E), their predecessor ISO/IEC 14882:1998(E) and technical corrigendum ISO/IEC 14882:2003, or language use that is classified as giving rise to unspecified, undefined or implementation-defined behavior.

SUPPORTED HOST PLATFORMS
- Windows 7 (32 and 64 bit)
- Linux RHEL 5 and above (32 and 64 bit)

IDE INTEGRATIONS
- Microsoft Visual Studio™ 2010, 2012, 2013 and 2015
- Eclipse V 3.5.2 and above
- Eclipse based IDEs

CONTINUOUS INTEGRATION ENVIRONMENTS
- Jenkins
- Other CI environments can be integrated through command line interface

SUPPORTED COMPILERS
- GNU gcc, g++
- MinGW gcc, g++
- Microsoft Visual C++
- Analog Devices VisualDSP++
- Altera Nios II gcc
- GCC ARM Embedded
- ARM RVCT
- Freescale CodeWarrior
- eCosCentric
- Green Hills C/C++
- IAR C/C++
- Keil
- TASKING VXToolset
- Texas Instruments
- Wind River Diab
- XILINX C/C++
- Other compilers may also be available

SUPPORTED VERSION CONTROL SYSTEMS
- AccuRev
- Clearcase
- CVS
- Git
- Mercurial (Hg)
- MKS
- Perforce
- PVCS
- Subversion
- Synergy
- Team Foundation Server

SGS-TÜV SAAR CERTIFIED

SGS-TÜV Saar has certified QA·C and QA·C++ as “usable in the development of safety related software” for the key safety critical standards, IEC 61508, ISO 26262, EN 50128, IEC 60880 and IEC 62304, enabling our customers to achieve product certifications to these standards more easily and in less time.

SGS TÜV SAAR FUNKTIONALE SICHERHEIT GEPRÜFT FUNCTIONAL SAFETY APPROVED

QA Systems and Programming Research Ltd

QA Systems is an authorised reseller of the QA·C / QA·C++, QA·Verify static testing tools and their compliance module add-ons, which are owned by Programming Research Ltd.

QA·C, QA·C++ and QA·Verify, offer the closest possible examination of C and C++ code. All contain powerful, proprietary parsing engines combined with deep accurate dataflow which deliver high fidelity language analysis and comprehension. They identify problems caused by language usage that is dangerous, overly complex, non-portable or difficult to maintain. Plus, they provide a mechanism for coding standard enforcement.

Contact Us

For further information regarding QA·C, QA·C++ and QA·Verify and compliance module add-ons, please contact QA Systems at **info@qa-systems.com** where appropriate QA Systems will re-direct you to Programming Research Ltd.