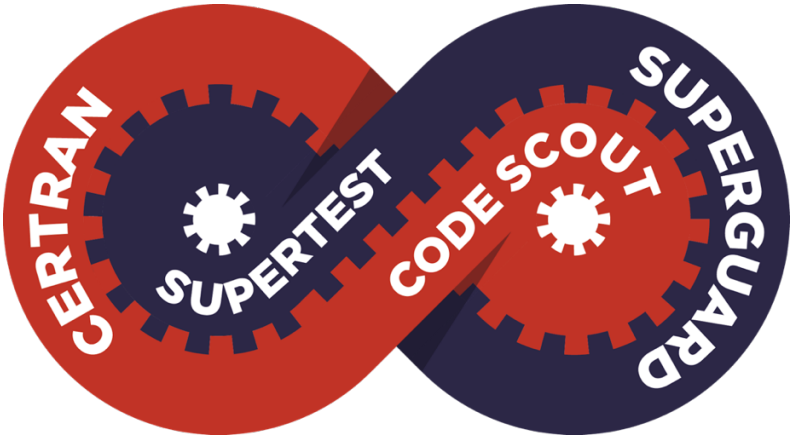
Continuous Qualification

Keeping Compilers, Libraries and Applications Inside the Safety Envelope

Roberto Bagnara, Ph.D. FUSA Manager, BUGSENG, roberto.bagnara@bugseng.com
Dr. Marcel Beemster, CTO, Solid Sands, marcel@solidsands.nl

Abstract

In safety-critical software development, qualification is not a one-time event. Yet in practice, compiler and library qualification is often treated as something that happens once, at the beginning of a project, and is disconnected from future project development. It is then archived as a PDF that everyone hopes will remain valid forever. This approach does not align with the realities of modern software development and internal safety standards, nor with the expectations of functional safety standards, regulators, or auditors.
What is needed instead is a new mindset: ***Continuous Qualification.***



Qualification is About an Envelope, not a Certificate

When we qualify a compiler or a library, we are not qualifying the tool as such. We are qualifying specific versions of the compiler, a specific version and subset of the library, with specific option combinations. Together, these parameters define the safety envelope. The application (including its build system) can have confidence in the safety of the toolchain only as long as it remains within the defined envelope. The moment the application and its build system leave the envelope, the evidence no longer applies and we need to make adjustments.

Qualification Must Be Continuous

Modern software development constantly pushes toward the edges of the envelope. The toolchain is updated with new versions that offer features, fixes and security patches. The build system evolves as it responds to changing application requirements, integration of new (third-party) components, newly implemented features and performance feedback. Each change potentially steps outside the safety envelope. In a modern, responsive software development environment, many of these changes may be introduced automatically. **Continuous Qualification** can support these changes and alert us when attention is needed.

Continuous Qualification Means Two Things

Continuous Qualification has two complementary dimensions: **Continuous Requalification** to detect changes in the toolchain and its use, and (if possible) automatically qualify them; and **Continuous Conformance** to ensure that the toolchain stays true to its specification. Both are required and we can easily see how they support each other.

Compiler Qualification in Practice

Compiler qualification by validation is the most practical approach allowed by safety standards such as ISO 26262. Solid Sands' **SuperTest** can be deployed to systematically verify the compiler, and its results used for qualification. It is easy to integrate into CI systems, but when doing so, the user must configure it with the exact compiler use cases that are defined in the application build system. Modern systems may have tens of use cases. That is where **ECLAIR CerTran**, from BUGSENG, comes in. It analyses the build by intercepting compiler calls, understands the options used and reconstructs the actual compiler use cases. With this information, it automatically generates as many SuperTest configurations as needed and runs them to verify the toolchain for those use cases.

With this, compiler qualification is repeatable, scalable and CI-friendly. Of course, anomalies still need to be investigated, but after the initial analysis these are typically very minor.

Library Qualification

Libraries, especially C++ standard libraries, pose different but equally serious challenges. BUGSENG's **ECLAIR Code Scout** analyzes the application and its build system, and generates a list of the library functions that the application uses. Solid Sands' **SuperGuard** then runs requirements-based tests for many of the headers in the C++ standard library, with high library code coverage. This provides evidence of the library implementation's compliance with the subset determined by ECLAIR Code Scout. SuperGuard's reports provide full traceability between the C++ specification, requirements, test specifications, tests and results, which is essential information for library qualification.

In typical applications, only a subset of the library is used and qualified. The edge of this limited scope is non-trivial to identify for developers, and in part implicit due to C++'s syntactic sugar. Without tooling, it requires great diligence by the developers to stay on the right side of the envelope.

Staying Inside the Envelope: The Role of *ECLAIR Safety Sentinel*

As we have shown, one-time qualification is not sufficient. After the initial baseline is established, the system must automatically detect when the application or build process moves outside the qualified envelope and then either execute an automated requalification or alert that an explicit decision is required to extend (or not extend) the qualification scope. This is the purpose of **ECLAIR Safety Sentinel**, which complements SuperTest, SuperGuard, ECLAIR Code Scout and ECLAIR CerTran. ECLAIR Safety Sentinel can be integrated in the application's CI/CD system. Whereas ECLAIR CerTran and ECLAIR Code Scout are initially used to define the use cases and scope required by the application and its build system, ECLAIR Safety Sentinel uses them to verify that they stay inside it. It monitors both compiler options used in the build and the library scope used by the application. Additionally, since it is based on ECLAIR's static analysis platform, it can be configured to check the mitigations defined for compiler and library defects. It provides a continuous guardrail that goes beyond traditional coding guidelines checking. This is huge because it relieves the developer from having to keep track of all the constraints of the safety envelope.

A Closed Loop for Continuous Qualification

Together, the integrated BUGSENG and Solid Sands solution enables a closed loop: compiler qualification is provided by ECLAIR CerTran and SuperTest; library qualification is provided by ECLAIR Code Scout and SuperGuard; and ECLAIR Safety Sentinel adds mitigation checking and orchestrates these tools to provide continuous monitoring, evidence and requalification.

This turns qualification from a static, document-heavy process into a living, continuous engineering activity—fully aligned with modern CI/CD practices and the expectations of functional safety standards.

Continuous Qualification is Not Optional Anymore

Software complexity, toolchain complexity, and regulatory pressure are all increasing. The question is not whether qualification needs to be repeated, but rather how often and to what extent it can be automated. Continuous Qualification is the only sustainable answer. And it is only possible when compiler and library qualification, static analysis and build interception work together as a single, integrated system.

BUGSENG and **Solid Sands** are working together with organizations in the field to make Continuous Qualification practical. If you are currently qualifying (or requalifying) compilers and libraries for safety-critical software, and want to make this process scalable, repeatable, and CI-ready, get in touch to explore how SuperTest, SuperGuard, CerTran, Code Scout and ECLAIR Safety Sentinel can be combined to support your Continuous Qualification strategy.