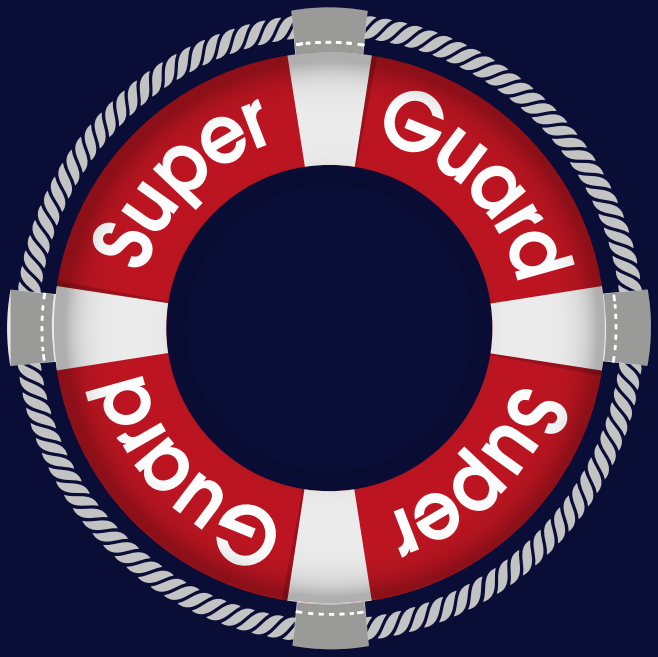


Library Safety Qualification Suite

Solid Sands

A world built on code needs Confidence by Design™

Solid Sands B.V. | Amsterdam | The Netherlands | www.solidsands.nl

The SuperGuard Library Safety Qualification Suite is a requirements-based test suite for the C and C++ standard libraries. It gives full traceability between the requirements derived from the ISO C and C++ language definition and the individual library tests. It is designed to support the qualification of implementations of the C and C++ standard libraries for safety-critical applications, both for third-party (COTS) and self-developed or maintained library implementations. SuperGuard provides a detailed breakdown of the ISO C and C++ library specifications into the requirements that must be met by an implementation of the library. These requirements are linked to test specifications that describe how a test verifies the requirements.

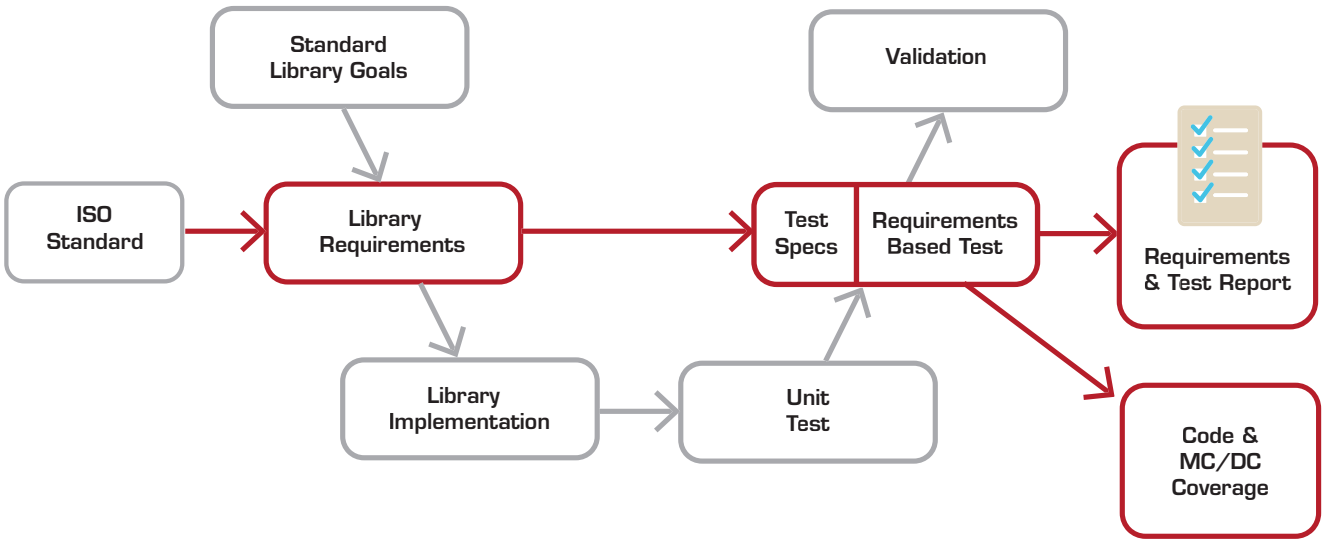
In turn, the test specification is linked to tests in SuperGuard. In this way, a detailed path is created from the specification to the tests that is easy to comprehend and verify, so that it can be used to create confidence in the compliance of the library implementation with the specification.

SuperGuard is built to achieve high structural code coverage of the target library implementation. In addition to requirements traceability, this provides a second path to demonstrate completeness of the test suite. The suite also includes a software tool that reports the requirements that are met, and which are not met, by a library implementation as verified by a run of the test suite for a specific use case or configuration.

Extended Overview

The tests in SuperGuard are based on the well-regarded and continually updated library test suites of SuperTest. In SuperTest, tests are organized according to the ISO C and C++ standard library specifications. For every section in the specification, there is a list of tests that verify that section. SuperGuard provides two, much more detailed, steps to bridge the gap between the ISO specification and the test suite.

First of all, we have analyzed the text in the library specification and turned it into testable requirements. The qualifier ‘testable’ is important here. Take for example the function ‘strlen()’, of which the argument must point to a valid string. That the argument is a valid string is a requirement, but it is not a requirement that is testable for the implementation of the strlen function. Instead, it is a requirement on the application that calls the function. It cannot be verified by a test of the ‘strlen()’ function. In effect, that requirement on the application becomes a pre-condition for the (testable) requirements on the implementation.



SuperGuard can be used to support the qualification of C and C++ Standard Library implementations for safety-critical applications, both for unmodified third-party (COTS) library implementations and self-developed or self-maintained library implementations. SuperGuard’s role in the V-Model for software development is shown in the image above.

In the specification, there are other details that do not translate into requirements and, details that implicitly do. With SuperGuard we have analyzed the text of the specifications and distilled the requirements on the library implementation that can be used to show compliance to the ISO specification.

The result of the first step is a detailed and structured list of requirements. The second step is the translation of requirements into test specifications.

A test specification describes how to test that the requirement is implemented correctly, and it is linked to specific test cases. If the requirement is composed of multiple cases, there can be multiple test specifications for a single requirement. This typically happens when considering boundary values and equivalence classes. Not all requirements can be turned into tests. That is because there is not sufficient information in the library specification. This happens for features of which part of the specification is left to the implementation. These are implementation-defined features. For example, the C specification allows for many different implementations of the 'locale' feature, and they are not defined in the ISO C specification. For such gaps in testable requirements, an application will have to refrain from using such implementation-defined features or will require additional feature-specific tests. If there is access to the sources of the library, the use of structural code coverage analysis further improves the confidence in the library implementation and the completeness of the test suite. We recommend using an existing tool for coverage analysis, those tools are readily available in the market. We have measured structural code coverage to reach 100% or close to 100% for many functions in the widely used library implementations, and also high figures for MC/DC coverage when using the SuperGuard test suite.

In addition to the requirements and test specifications, SuperGuard contains a reporting tool. The reporting tool can be used after completion of a library test run. It interprets the test results and links them back to the library implementation requirements. It provides a detailed overview of all the requirements that are met by the library implementation, and those that are not. A requirement may not be met because it is implementation defined as described earlier above, or because the test has failed.

SuperGuard Features

- Requirements-based testing for the C and C++ standard library
- High structural code coverage and MC/DC
- CSV report linking requirements, test specifications and test results
- Compliant with the requirements of functional safety standards

Test Platform Features

- Multi-processing speedup
- Supports bare-metal targets
- Requirements traceability
- Windows and Linux hosts
- Full control over test sets
- Python configuration plugin
- Reporting tools

Compliant With

- ISO 26262
- IEC 61508
- EN 50128 / EN 50716
- IEC 62304
- And many more