PERFORCE

WHITE PAPER

Getting Started with Compliance: Simple Steps for Critical Software Development Projects

Introduction

In this whitepaper, we take an in-depth look at the core fundamentals of software compliance.

An essential part of critical software development is ensuring that it is compliant with the necessary coding guidelines and other functional standards.

Here, we provide an overview of:

- What Is Software Compliance
- What Are Software Standards
- What Are Coding Standards
- What Are Functional Safety Standards
- What Are Functional Security Standards
- Best Practices for Software Compliance
- How to Enforce Continuous Compliance with Every Commit

What Is Software Compliance

Compliance can mean complying with a wish or command — so, it can mean a wish to have good quality software. However, in general, software compliance means conforming to a specific set of rules or standards. To be more specific, software compliance is a quality measure that provides an indication of how well your software meets the rules in a standard.

To understand software compliance, you must begin with software standards. These standards generally provide the rules to help ensure that software is safe and/or secure. If your software complies with software standards, it is less likely to have bugs, security vulnerabilities, or design flaws.

What Are Software Standards

Software standards are the general rules and best-accepted practices to provide high-quality software products or systems.

The main aim of complying with standards is to ensure that products, systems, and organizations are safe, reliable, and good for the environment. Standards help make systems interoperable and are important for effective communication between disparate systems.

Over the years, and across geographies and industry sectors (such as automotive, aerospace and defense, medical device, and rail), governments and industry groups have issued many software standards.

They can all be classified as belonging to one of the following categories:

- **Coding Standards**
- **Functional Safety Standards**
- **Functional Security Standards**

What Are Coding Standards

Coding standards are collections of coding rules, guidelines, and best practices that capture many person-years of expertise to basically define a language subset and provide a set of rules to follow.

Coding languages such as C and C++ have significant cases of unspecified and undefined behaviors that can be avoided by defining and using a language subset. Also, in some cases, different compilers may implement only a subset of the language standard or interpret its meaning in subtly different ways. This can lead to porting and semantic errors in the future. However, using a language subset can prevent you from adopting these more precarious constructs.

Most coding standard rules can be enforced through static code analysis, and by following them, the probability of introducing errors is greatly reduced. When good standards and coding best practices are not applied, code is more likely to be prone to bugs, contain security vulnerabilities, or be difficult to maintain.

Therefore, complying with coding standards leads to more consistent, readable, and understandable code that helps to reduce future maintenance efforts.

A coding standard may be specific to an organization or project or an accepted industry standard such as MISRA, OWASP, or CERT.

In addition to rules based on the coding language, there are also coding style guidelines that help provide consistency. This is key for improved readability and maintainability of the software.

The major coding standards have specific requirements for measuring compliance but they often include:

- Disciplined software development process, which includes a selection of tools.
- Code that has shown not to contain any violations of the rules specified in that standard.
- Authorized and documented exceptions.

Showing compliance to a coding standard should be part of any software development lifecycle and will:

- Increase the efficiency of the software and reduce the development time.
- Help in detecting errors in the early phases, so it helps reduce the overall cost incurred by the software project.
- Increase readability and clarity — thus reducing the complexity of the code.

What Are Functional Safety Standards

Functional safety standards have been developed to prevent hazards and potential human health and life-threatening failures by minimizing the risks associated with product design and development. The goal is to achieve acceptable residual risk.

Systems covered under functional safety are designed to automatically prevent dangerous failures or to control them when they occur. This is done by producing systems that can execute specific functions correctly, even under non-intended use or sometimes even misuse.

These process standards guide the whole product safety lifecycle: management, development, production, and service. Throughout the development process, the standards cover all safety-related aspects on a very detailed level, from requirement specification through design, implementation, and validation, right through to decommissioning.

Complying with a functional safety standard means that processes have been specified for all requirements in the standard. These processes can be followed throughout the supply chain so that information about the system can be communicated using common terminology and system parameters.

Functional safety standard compliance makes the development and production processes more effective and structured. Although more effort is required, the result is an organized process that ensures any possible weak points are identified and addressed. This results in a safer and higher-quality product.

Each industry typically has a standard to guide development and set minimum expectations.

Most of the standards in common use are derived from IEC 61508, which is an international standard for electrical, electronic, and programmable safety-related systems. It sets out the requirements to ensure that all elements of the product lifecycle consider safety.

The most well-known derivatives of the IEC 61508 standard are:

- ISO 26262: Automotive
- EN 50128: Rail
- ISO 62304: Medical Devices
- ISO 60880: Nuclear Power Plants

Additionally, there are equivalent aerospace standards DO-178B/C that specify the functional safety requirements for software used in airborne systems.

Many functional safety standards recommend the use of a coding standard and often a language subset, for example, ISO 26262 gives the example of MISRA C for C code.

What Are Functional Security Standards

In the same way that functional safety standards encourage the development of systems with safety in mind, functional security standards require that security is at the forefront of development considerations from the start to build a culture of ownership around security.

Functional security requirements describe functional behavior that enforces security and can be directly tested and observed to ensure that developers display due diligence when it comes to the security of their software. This is essential as security vulnerabilities allow the software to be abused in ways that the developers never intended.

The use of security-focused coding standards — such as MISRA, OWASP, and CERT — helps drive the "built-in

security" of the software components and ultimately the devices, which leaves them less prone to external attacks.

There are several standards that are related to functional security:

- **ISO 21448 – Safety of the Intended Functionality (SOTIF):** The basic principles of SOTIF are identifying and minimizing risk, using safe and secure design principles, and — although there are no specific software requirements — employing an established software development system.
- **ISO 21434 Road vehicles — Cybersecurity engineering:** ISO 21434 focuses on the cybersecurity risk in road vehicle electronic systems and covers all stages of a vehicle's lifecycle.
- **UNECE WP.29:** Vehicle type approval and cybersecurity management systems that define safety and environmental performance requirements.

Both ISO 21434 and the UNECE WP.29 regulation require threats and risks to be analyzed throughout a vehicle's lifecycle to determine the extent to which a user can be impacted by a threat scenario. Compliance with ISO 21434 can be used to demonstrate compliance with the UNECE WP.29 regulation. In addition, ISO 21434 requires the use of a coding guideline – for example, MISRA C or CERT.

Best Practices for Meeting Software Compliance Requirements

There are six best practices that you should follow when meeting compliance requirements for software standards.

1. KNOW WHY YOU'RE USING THE CODING STANDARD

The purpose of a coding standard is to restrict the use of problematic areas of the programming language, prevent undefined or unspecified behavior, and limit the use of error-prone constructs.

In addition, using a coding standard improves the readability, maintainability, and portability of your embedded software code.

2. CHOOSE THE BEST CODING STANDARD FOR YOUR PROJECT

The coding standard that you use must be appropriate for your project. Often, the industry associated with your project will help clarify which coding standards you should use.

For whatever standard you use, it should be recognized and accepted by the industry. This is an extremely important step because using an industry-recognized standard reduces the amount of qualification work needed to satisfy your end-users.

A good example of an industry-recognized guideline is MISRA, which was written for the automotive industry but is now used for all safety-critical applications.

3. USE CODING RULES AND FOLLOW RECOMMENDATIONS

Coding standards often include two elements — rules and recommendations (or directives). Rules are considered the backbone of a coding standard while recommendations (or directives) are usually advisory. Most coding standards have both rules and recommendations.

Compliance is normally defined in terms of rules. Modern rules are often stated in such a way that makes it easy to verify them with a static analysis tool. For example, the rules are "statically enforceable".

Normally, recommendations (or directives) do not impact compliance and are usually not statically enforceable.

4. ASSESS CODE QUALITY AND PRIORITIZE BUG FIXES

When you check your code against the standard, you may get a list of compliance errors, bugs, and defects. Defects need to be prioritized so the high severity ones are fixed first.

To effectively address this list, you should categorize the rules with a severity matrix to help you better assess code quality and prioritize fixes. Evaluating the quality of your code can help you to decide whether or not you can move into production.

One of the most efficient solutions for assessing code quality and prioritizing bug fixes is to use a static analysis tool.

5. PLAN FOR RULE DEVIATIONS

Rules do not fit all situations and there will always be deviations. There are three key factors for planning for rule deviations.

1. Decide which guidelines can be deviated from and which cannot.
2. Specify how a deviation can be implemented.
3. List what needs to be documented or recorded when a deviation takes place.

Having a plan for rule deviations will help you if you need to provide evidence to an auditor.

6. EDUCATE YOUR TEAM

One of the central goals of any coding standard is to educate, and supporting materials for a standard can help provide that education. Supporting materials help provide the background or provide extra information. For example, this might include links to documentation — such as ISO standards.

How to Enforce Compliance with Every Commit

Compliance with any standard should be implemented from the start of any software project and throughout its life cycle. This ensures that there are no nasty surprises at the end of the project and that the software is of high quality.

Using a static analysis tool — such as Helix QAC or Klocwork — as part of the development process, makes compliance to coding standards simple. What's more, a static analysis tool is able to support continuous compliance to ensure that each component is compliant at every stage of development.

Static analysis can provide even more benefits, including:

- Detection of these software compliance issues as early as possible in the development lifecycle.
- Promoting a shift-left approach to the compliance process.
- Accelerating and automating code reviews.
- Reporting compliance over time and across product versions.

Compliance with the necessary coding guidelines and other functional standards is an essential part of critical software development, and the use of a static analysis tool makes that process easier and more efficient.

See for yourself how Perforce static analysis tools help ensure that your embedded software is compliant.

Request your free trial today.

STATIC ANALYSIS TOOLS FREE TRIAL

perforce.com/products/sca/free-static-code-analyzer-trial

About Perforce

Perforce powers innovation at unrivaled scale. Perforce solutions future-proof competitive advantage by driving quality, security, compliance, collaboration, and speed – across the technology lifecycle. We bring deep domain and vertical expertise to every customer, so nothing stands in the way of success. Privately held and funded by Clearlake Capital and Francisco Partners, our global footprint spans more than 80 countries and includes over 75% of the Fortune 100. Perforce is trusted by the world's leading brands to deliver solutions to even the toughest challenges. Accelerate technology delivery, with no shortcuts. Get the Power of Perforce.