Accelerating MISRA C and SEI CERT C Compliance with Dedicated Reporting and Workflow Management

Coding standards can be used to make source code better, more portable, readable, and predictable. The most intensive use of coding standards are in safety-critical industries, where safety and security are the focus. Coding standards compliance is also explicitly or implicitly required by many functional safety standards (for example, ISO 26262 explicitly recommends static analysis and coding standards compliance, mentioning MISRA as an option).

This paper discusses what it really means to achieve coding standard compliance, using MISRA C 2012 and SEI CERT C as examples, and how to accelerate compliance with tool automation, dedicated reporting, and workflow management. The recommendations made here are generic and can be applied to any coding standard.

HOW DO YOU CHOOSE A CODING STANDARD?

There are many coding standards to choose from – MISRA, SEI CERT, AUTOSAR, JSF, UL 2900, and others. So, the natural question arises: Is there one good coding standard to use to achieve safety and security? Perhaps the project needs to comply with more than one coding standard? If so, which ones? The common sentiment in the industry is to associate certain coding guidelines with a specific goal for code compliance, such as safety or security.

Generally speaking, MISRA and AUTOSAR C++ are specific for improving safety, while SEI CERT is used for improving security. But just as safety and security overlap in the real world, there is a significant overlap between these standards. Taking a closer look at MISRA C 2012, for example, it covers security aspects quite well. Nonetheless, CERT has broader support for security and treats this subject more thoroughly.
Does it make sense to attempt to comply with more than one coding standard? In general, it's not recommended to strive for compliance with more than one standard. Mainly because of the overlap, which results in wasted effort and a reduced return on investment of adopting the standard and associated tools. However, it may make sense to pick one primary standard as a base and extend it with selected guidelines from other standards that make sense for the project. For example, if the project requires compliance to functional safety standards, the project could use MISRA as the base and extend the core set of rules with some additional rules from CERT.

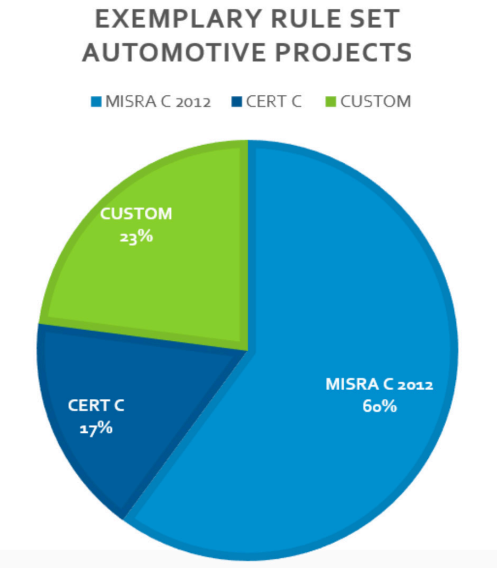

Figure 1: Parasoft observed distribution of coding standard rules for automotive projects.

Parasoft's experience with its customers building safety-critical automotive software, illustrated in Figure 1, shows this mixed approach. These customers selected MISRA C 2012 as a base standard for their development and supplemented it with a selection of the rules from CERT and a collection of custom guidelines developed internally in the organization (created using Parasoft C/C++test extension tools for custom checkers). The customers settled on a rule set composed of 60% MISRA rules/directives, 20% CERT rules and 23% custom checks, for a total of 250 rules.

PARASOFT®
Automated Software Testing

For the best results, an orthodox approach of adopting only one standard should be avoided in favor of a hybrid approach with one main standard and supplementing it with additional selected guidelines. Selecting these guidelines is not the focus of this paper but the discussion of a pragmatic approach to adopting standards is part of an overall approach to reducing the impact and cost of these standards.

WHAT DOES IT MEAN TO BE COMPLIANT?

The strict definition of compliance is, to some extent, loosely defined in the industry. What does it mean to be MISRA or CERT compliant? How does an organization prove to auditors due diligence and adherence to the spirit and letter of the each guideline?

Many organizations have their own definitions of compliance, based on general principle that the source code is free from violations of the coding guideline. At a high level, this seems sufficient, unfortunately the imprecise definition of compliance is very often a reason for friction between manufacturers, sub-contractors, certifying authorities, and customers. A clear recommendation Parasoft makes to all of its customers is to have a clear definition of compliance to a specified standard if you are developing a software for external customer. The definition of compliance should be an inherent part of business negotiations and be precisely defined. Luckily, in some cases, the standards themselves provide such guidance.

MISRA COMPLIANCE REQUIREMENTS

In response to ambiguity about compliance requirements, a dedicated document titled MISRA Compliance 2016: Achieving Compliance with MISRA Coding Guidelines was created to clarify requirements. This document precisely defines how to achieve compliance and what kind of documentation shall be prepared to prove achieved compliance. At the root of this document is the assumption that the software is developed with the disciplined and documented development process, with compliance activities integrated from the very beginning. So, we need to scan the code for compliance with guidelines and maintain appropriate documentation of the process MISRA standard mentions four main reports from compliance process:

- Documentation stating how the guideline is going to be enforced, the Guideline Enforcement Plan
- A record of any changes to the default categories of MISRA rules, the Guideline Re-categorization Plan
- The documented deviation process with deviation records and deviation permits, the Deviations Report
- A document summarizing the project compliance level, the Guideline Compliance Summary

Not only must team leaders make sure developers create the source code free from violations of MISRA guidelines, but they also need to assure documentation of the process and generate very specific reports.

SEI CERT C CONFORMANCE

The SEI CERT C coding standard does not require a lot of documentation to be prepared to claim compliance. Quoting the standard:

> "Conformance to the CERT C Coding Standard requires that the code not contain any violations of the rules specified in this standard. If an exceptional condition is claimed, the exception must correspond to a predefined exceptional condition, and the application of this exception must be documented in the source code."

Note that conformance is based on rule violations, CERT C distinguishes between rules and recommendations. Rules are considered to be obligatory while recommendations are not. Exceptions made for rule violations must be documented. CERT C also classifies rules and recommendations in the standard into three levels based on a risk assessment that considers the severity, likelihood, and remediation cost. Rules classified as L1, for example are high severity, likely to occur and inexpensive to repair. Conversely, where L3 represents rules that are low severity, unlikely to occur and expensive to fix. See the following diagram from the standard:

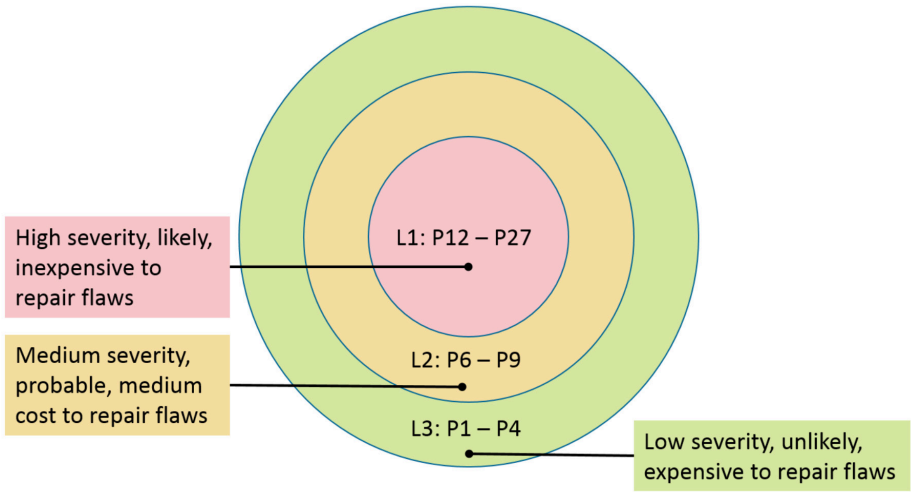

Figure 2: SEI CERT C rule classifications based on risk assessment.

Conformance to CERT C is based on these levels. Software can be claimed to be L1, L2, or full conformant based on what class of rules are met. This is a useful specification because it allows developers to focus on the most critical security practices first and achieve the highest return on investment for implementing the standard.

Deviations from the rules is necessary in very specific cases. If automated detection, e.g. with static analysis tools, finds a rule violation but the code is indeed correct, this exception must be documented. However, deviations are not granted for performance or usability reasons, and are at the discretion of the lead assessor. If enough evidence is provided that a vulnerability doesn't exist, exceptions are accepted. In practice, it's often easier to fix the code to conform to the rule that evidence that a vulnerability doesn't exist.

Accelerating Coding Standard Compliance with Workflow Management

Zooming in on a typical developer workflow during active software development, the process consists of coding (either new code, refactoring, or fixing existing code), local unit testing, submitting code to source control, initiating a continuous integration (CI) build and receiving feedback from such a build, fixing errors, and continuing on to the next function to implement. Introducing a coding standard into this day-to-day process is time-consuming and intrusive. It's no wonder that many of the industry standards highly recommend automated static analysis to help enforce and document coding standard compliance.

Consider a revised day-to-day workflow that incorporates a coding standard (we are using MISRA C and CERT C as our examples here):

1. A team lead, architect, or functional safety officer defines the test configuration with a collection of static analysis checkers to enforce the coding standard. This may be a one time process or repetitive action over the project lifetime.

2. The initial configuration that is shipped with the solution can be adapted for the needs of specific organization. This configuration tends to be a combination of rules from MISRA C, CERT C and custom rules. Parasoft C/C++test supports multiple configurations that are customizable and shareable to the entire team.

3. Most coding standards allow for customizations, the configuration should consist of rules the development team agrees are mandatory, at least initially. Rules considered advisory or recommendations can be disabled in the configuration.

4. Once the test configuration is prepared it is automatically made available to all the team members, directly in their IDEs, for continuous usage during software development.

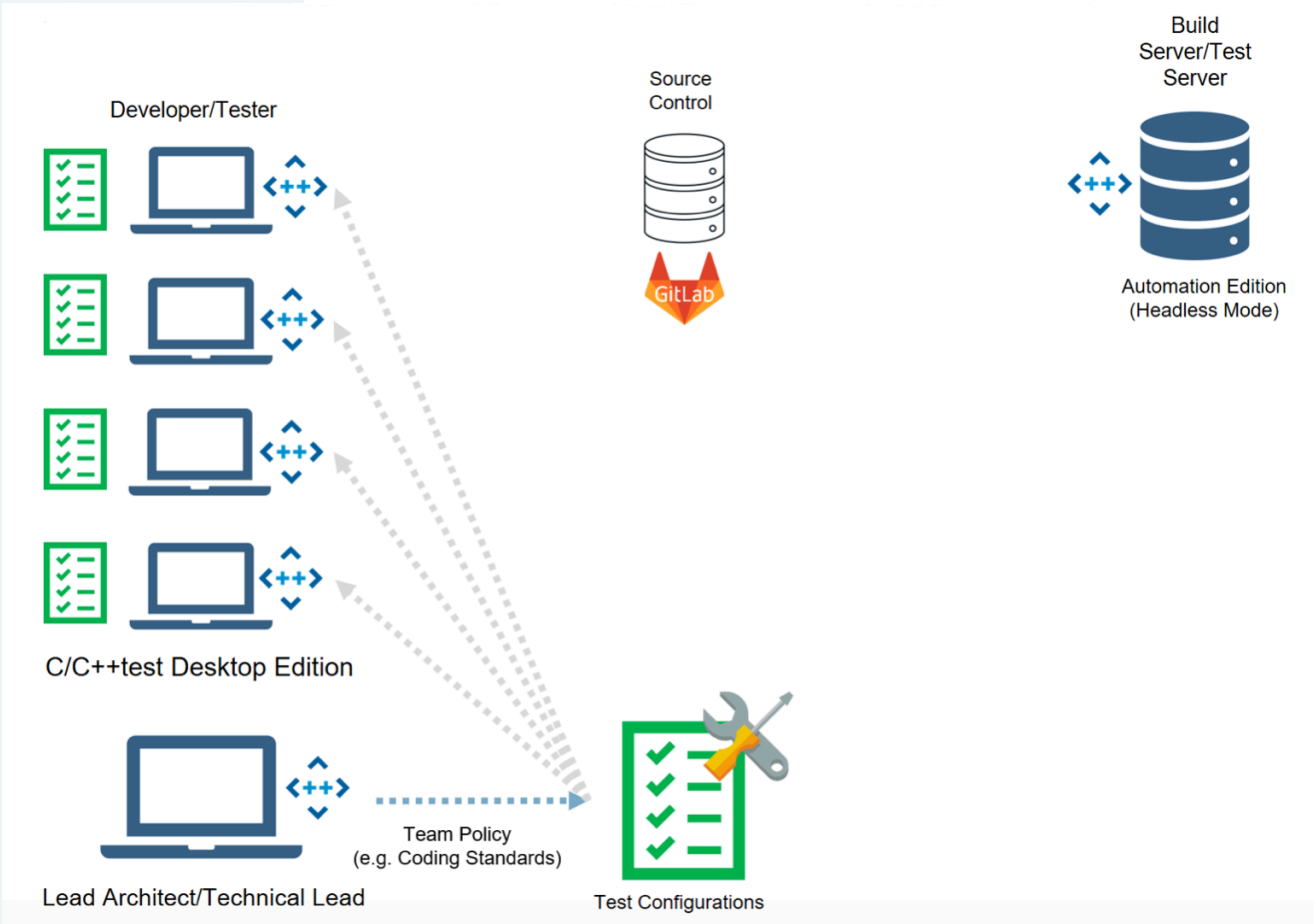

Figure 3: Technical lead or architect creates the team policies via a test configuration in Parasoft C/C++test and distributes to the team.

5. This point in the workflow is critical in order to accelerate the compliance process and take advantage of the benefits of early defect and security vulnerability detection of automated static analysis. Developers scan their code right after it is created, near instantaneously. From our Parasoft's observations, even 10 to 20 minute delays in delivering the compliance scan results is long enough for developers to lose focus and continue with other work.

6. In the next step developers check-in their code, what triggers the CI build, where an additional compliance scan is made. The questions often arises if it makes sense to set up static analysis results as a gate for the code check-in -- if the source code is not compliant then the check in is rejected. In Parasoft's experience this does not work well. Developers get easily frustrated by rejected commits and team work is hindered, and dependent pieces of code are not integrated on time. Parasoft recommends workflows do not block code check-ins but, rather, assume that any violations that make into the source repository are caught at the CI level.

7. During the CI build, a full scan of the source repository is made. Why perform an additional scan if the code is already scanned in the IDE? Integration-level scans provide a safety net which is required since some guidelines are detectable only at the system level, or a violation is simply overlooked. Also, a full system view of the source is needed for more complex static analysis to help detect defects and security vulnerabilities.

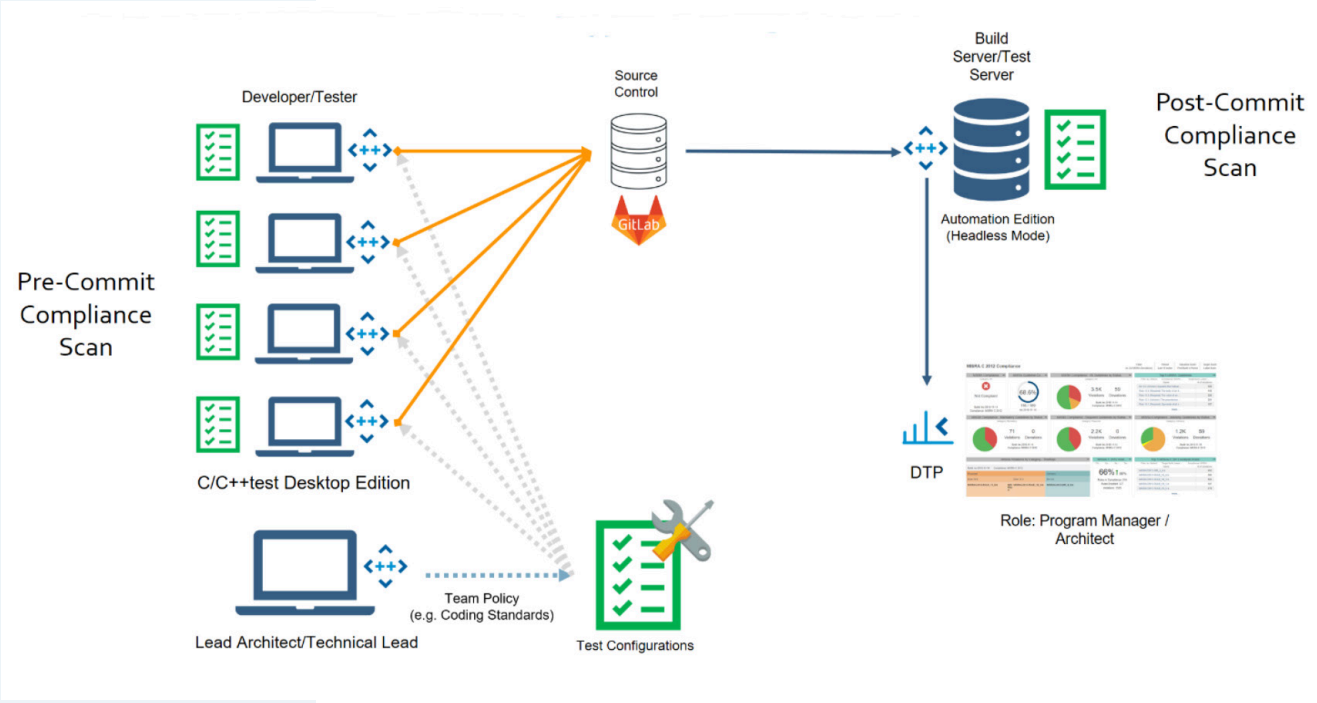

Figure 4: Developers scan their code for compliance before committing their code to source control. This early detection prevents most violations and defects from entering the build.

8. The results from the CI scan are published to Parasoft DTP which stores and analyzes the data.

9. Team leaders can use the web portal to access the Parasoft DTP results to discover the current state of compliance and drill down into specific areas of concern. They can then assign tasks to the developers to follow up on violations found during the analysis.

10. Developers then fix these problems, scan the code locally and commit corrections initiating another cycle.

11. As the project gets close to completion and the team is close to its compliance target, compliance reports are automatically generated, including all the documents that are required by the primary coding standard that is in use. These dedicated reports, specific for the standard, are a huge time saver, reducing the amount of tedious manual work related to creating and maintaining the compliance documentation.

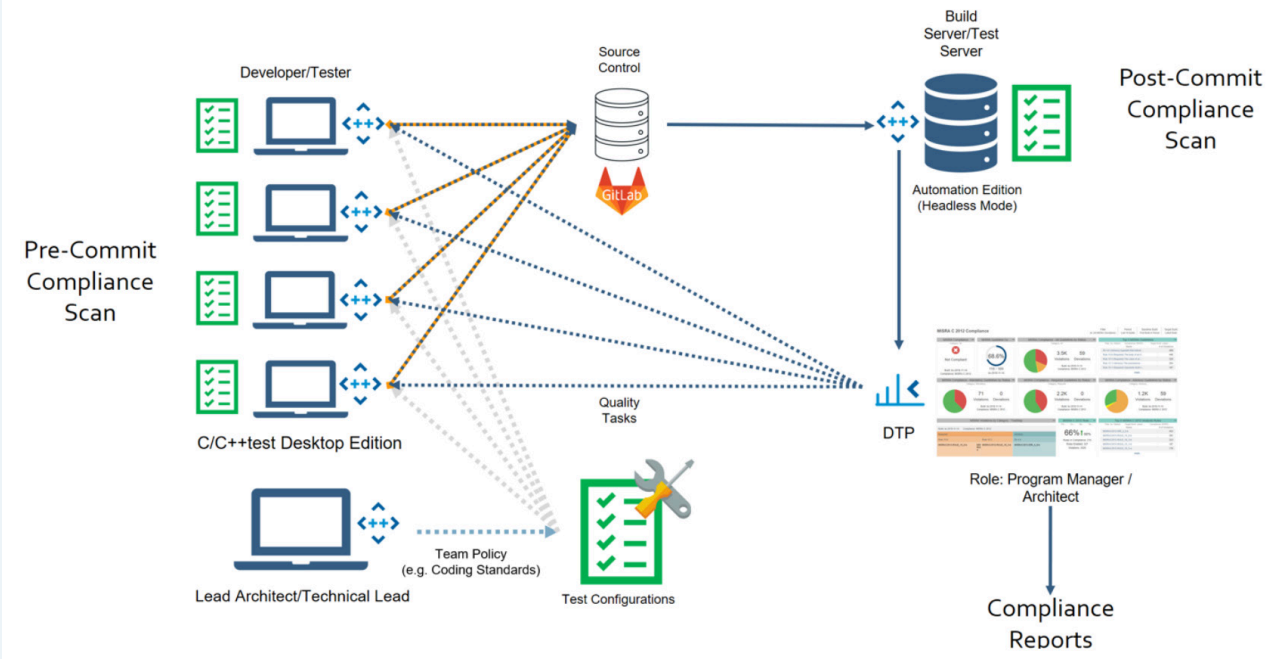

Figure 5: The source code commit initiates a CI process which includes a full scan of the project source. Catching more sophisticated violations and defects. Results are stored and analyzed by Parasoft DTP.

MISRA Compliance Reporting

Parasoft C/C++test provides dedicated reporting for documenting compliance to MISRA C. A dashboard on the Parasoft web portal provides at-a-glance views on the current state of the project. An example is shown below. Each of these dashboard widgets is linkable to a more detailed view such detailed violation reports, files and source code.

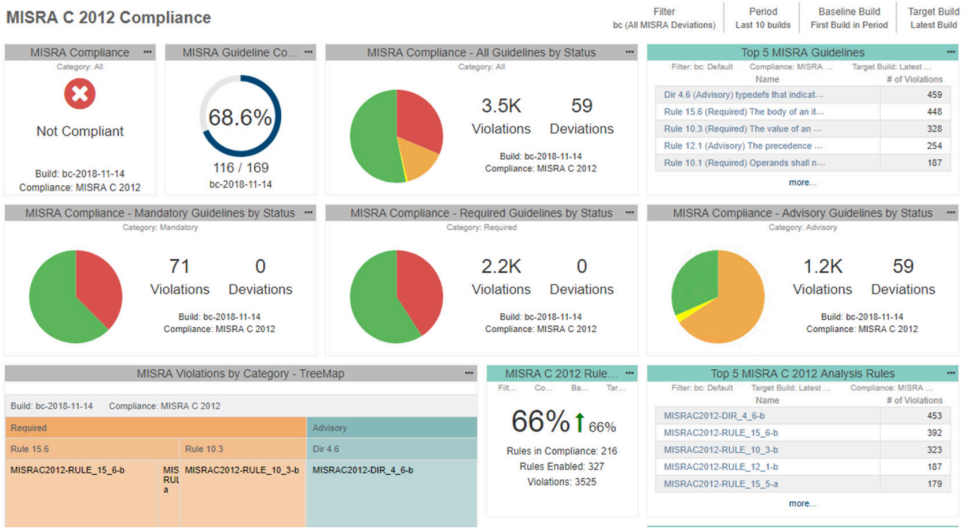

Figure 6: Parasoft MISRA C compliance dashboard.

Parasoft C/C++test provides the necessary reports needed to document MISRA compliance as outlined in MISRA Compliance 2016: Achieving Compliance with MISRA Coding Guidelines. Automating these reports is a big time saver, greatly reducing the amount of manual work required to document project compliance.

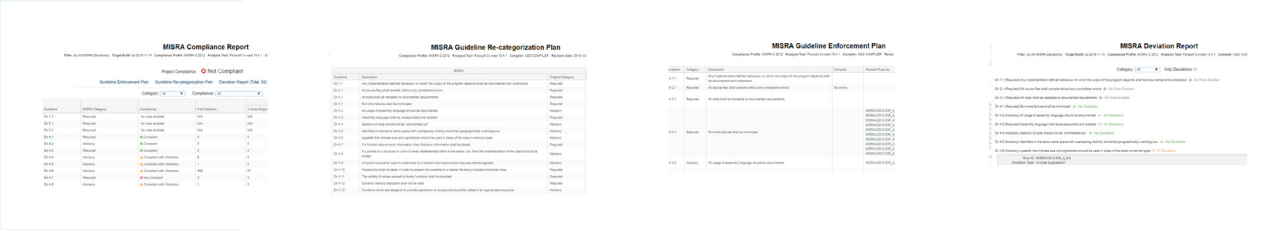

Figure 7: Parasoft C/C++test dedicated reports for MISRA C compliance

SEI CERT C Conformance Reporting

Although the SEI CERT C standard doesn't require specific compliance reports it does require a project to document conformance to the rulesets (e.g. L1, L2 and fully compliant.) Parasoft C/C++test includes a dedicated dashboard for CERT C conformance, as illustrated below.

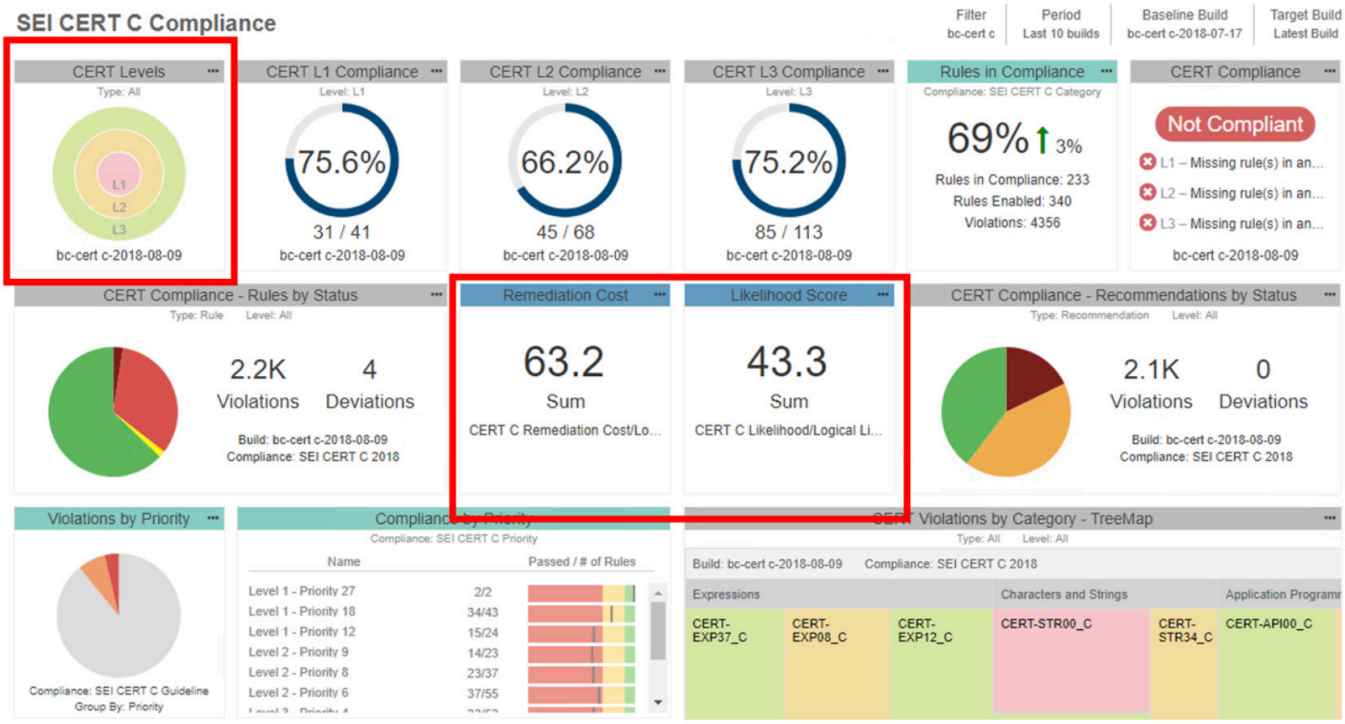

Figure 8: Parasoft CERT C compliance dashboard.

Team leads can use this dashboard view to dig deeper into specific areas of concern and assigned tasks to developers to increase conformance over time. Viewing the results in context of the risk assessment framework used by the coding standard itself (for instance, seeing specific violations of L1 guidelines), significantly streamlines the process. Automating this reporting reduces the amount of analysis team leads and architects need to perform in order to achieve CERT C conformance.

SUMMARY

Coding standards are designed for software developers creating safety and security critical applications. Although many different standards exist, software teams are encouraged to create a customized set of rules that best fits their primary needs. Safety critical projects may emphasize a standard such as MISRA C, but also include critical rules from a security standard such as SEI CERT C. To accelerate compliance, two elements are essential: short feedback loops for developers working with IDEs and CI/CD scans for process management and reports generation.

Automation via static analysis is key to not only achieving compliance/conformance to the rule set but also reducing the manual effort of documenting and reporting compliance to auditors and assessors. The approach recommend in this paper includes an augmented developer workflow that leverages Parasoft C/C++test to help assess and monitor compliance on a daily basis, with dedicated reporting tools to help teams manage and achieve compliance.

ABOUT PARASOFT

From development to QA, Parasoft's technologies reduce the time, effort, and cost of delivering secure, reliable, and compliant software, by integrating static and runtime analysis; unit, functional, and API testing; and service virtualization. Powerful reporting and analytics help users quickly pinpoint areas of risky code and understand how new code changes affect their software quality, and groundbreaking technologies that add artificial intelligence and machine learning to software testing make it easier for organizations to adopt and scale an efficient software testing practice across development and testing teams.

www.parasoft.com

Parasoft Headquarters:
+1-626-256-3680

Parasoft EMEA:
+31-70-3922000

Parasoft APAC:
+65-6338-3628

PARASOFT®
Automated Software Testing