

How to Review Your Code Quickly and Efficiently

Explore the benefits of using Polyspace static analysis for your code reviews.

Introduction to Code Reviews

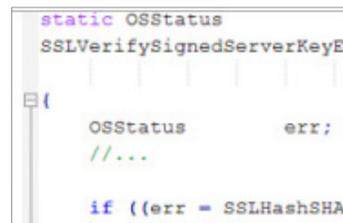
Code reviews: What are they, and why are they so vital? More importantly, do they always have to be time consuming?

During a code review, developers compare their designs against their requirements and identify defects in the implementation. It is an important verification step in the development of critical embedded systems, especially those that require certification.

Engineers are aware of the importance of code reviews, but the process can be time consuming, tedious, inefficient, and therefore ineffective. More significantly, code reviews often devolve into bug-finding expeditions, taking time away from the review's key task: improving the software design.

Human Code Reviewers Are Fallible

In 2014, technology blogs and other outlets were abuzz with the news of a vulnerability in devices developed by a major software company. There was a problem in the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) code. This code could be exploited by what is known as a Man in the Middle attack (MitM). This was a simple issue that developers should have detected during a code review.



```
static OSStatus
SSLVerifySignedServerKeyE
{
    OSStatus      err;
    //...
    if ((err = SSLHashSHA
```

A good software development workflow includes peer code review as part of the verification process. However, as mentioned previously, the process can be time consuming and tedious, as code reviews often become style checking and bug finding expeditions rather than a discussion aimed at improving the software design.

Therefore, a code review focused on scanning the code for defects is highly ineffective, because detecting subtle run-time errors, or even simple coding defects, by manual review is challenging. As in the case above, an undetected bug can slip into production code.

Then there is the challenge of finding a balance between the time spent reviewing code versus the time spent on other verification activities, given that the optimal rate of reviewing code is of the order of 100 LOC/hour. How long would it take an engineer to review an average software module, which stands at about 10 KLOC?

What Makes a More Efficient Code Review?

An effective code review is one focused on design aspects such as meeting the design requirements, identifying missing requirements, and reviewing the design architecture and design of the interfaces. Static analysis can augment this process by automating some of the more tedious tasks like checking for compliance with coding guidelines or detecting different categories of defects. The artifacts and metrics generated from these tools provide a basis for a more evidence-based peer review process.

Three Benefits of Using Polyspace for Your Code Review

1. Ensure code compliance

Developers commonly use static analysis to check code for compliance to coding standards. Static analysis tools such as Polyspace® help to automate and streamline these activities ahead of a code review process. Using a static analysis tool helps to ensure that your code is compliant with coding standards such as MISRA®, JSF®, or your own custom coding rules.

2. Find bugs and security vulnerabilities early

What makes Polyspace stand out from the competition? In addition to coding rules, Polyspace tools detect a wide category of software defects and security vulnerabilities as part of the coding process. You can address most of these defects before you review the code with your team. This saves you time and helps the review team focus on important things like software design and requirements. Furthermore, defects such as concurrency and static and dynamic memory issues are complex and often missed by the manual review process. Polyspace helps you automate the process of finding them. When you can prove absence of errors, you can reduce the time spent reviewing good code.

3. Understand how your code will run

With Polyspace, you get detailed information regarding the control and data flow through your code in the form of the possible variable ranges. You get function call graphs and data dictionaries to show where variables are written to and read from. This information can be invaluable during a code review where it might be important to understand the run-time behavior of the software. This helps you make better design decisions.

The process is scalable: Analyze a few hundred lines of code, or analyze a few hundred thousand.

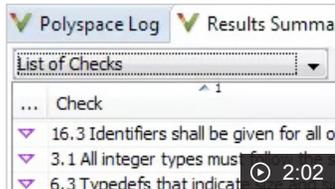
Explore How Other Engineers Use Polyspace



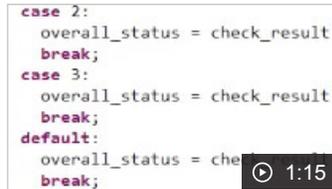
Solar Impulse Uses Polyspace Static Analysis for Solar Airplane

Learn to Integrate Polyspace into Your Workflow

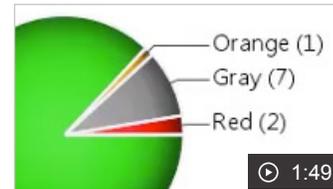
Watch the videos to learn how you can integrate Polyspace into your workflow.



Software Development Workflow: Enforcing Coding Rules



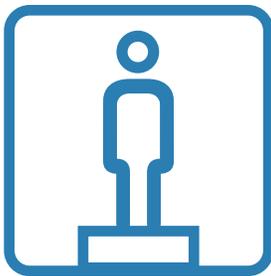
Software Development Workflow: Finding and Fixing Bugs



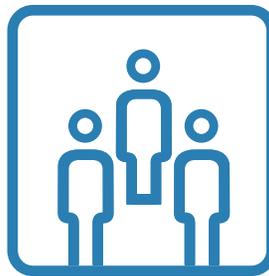
Software Development Workflow: Verify Absence of Errors

Interested in Learning More?

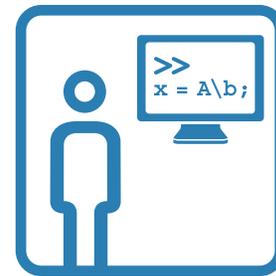
Select your role to explore how Polyspace can help you.



Software Development Manager



Software Engineer or Developer



Test or Quality Engineer

Request a Trial | Speak to an Expert