MathWorks
AUTOMOTIVE CONFERENCE 2023
North America

Find C/C++ Bugs as You Code
Integrate Polyspace into Your IDE

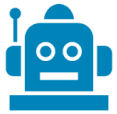
John Boyd, MathWorks

MathWorks®

Comprehensive static analysis tools for increased efficiency

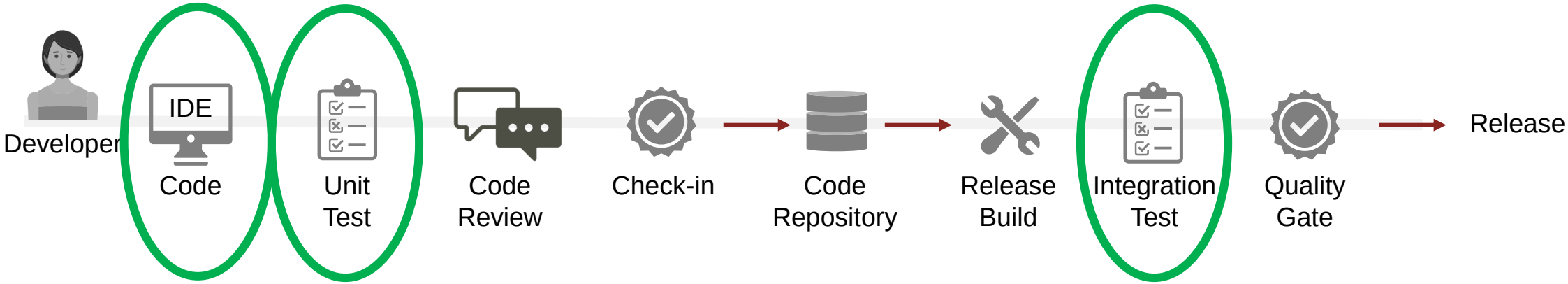
Catch and fix bugs while you code

Automate with CI Workflows



Collaborate with Team Members

Leverage static analysis components at critical points in the workflow

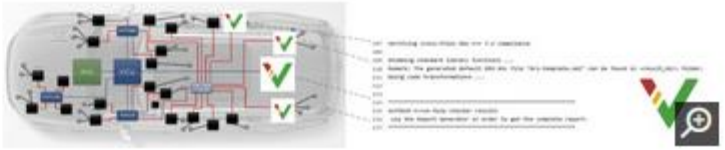


1. Repeatability
2. Faster delivery
3. Higher quality

Successful use of tool suite at Volvo

Volvo Cars Software Factory Increases Pace and Quality of Development with Polyspace

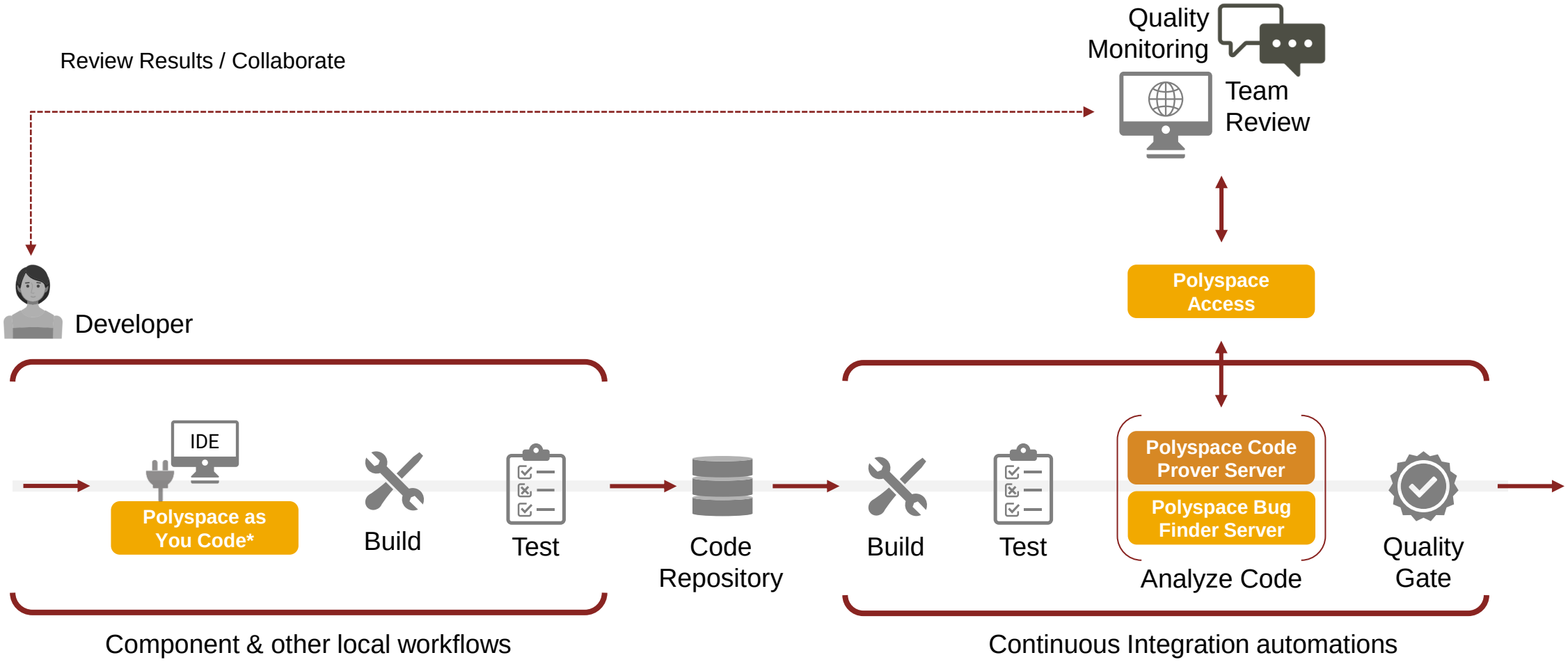
> "With Polyspace, we can ensure software security and quality by identifying and fixing critical run-time errors before every code merge."
>
> — *Johannes Foufas, Volvo Cars*



Volvo Cars uses Polyspace for static code checking throughout the development lifecycle.

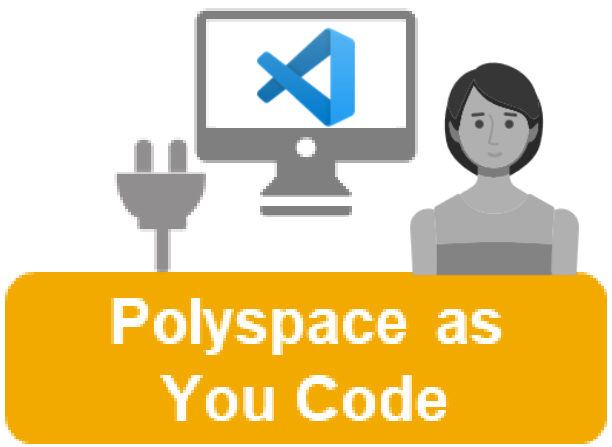
With the Polyspace as You Code plugin available in Polyspace Access™, several teams check adherence to CERT® C, CERT C++, MISRA C®, and AUTOSAR C++14 coding guidelines while they are coding in their IDEs. Before submitting their code modifications, developers run Polyspace Bug Finder™ and Polyspace Code Prover™ on their local computers to prequalify their changes.
When developers push their changes to the source code repository, it automatically triggers Polyspace Bug Finder Server™ and Polyspace Code Prover Server™ analysis. The Polyspace results are integrated into Gerrit to support code reviews. The CI system employs strict gating: every proposed change is verified before a code merge and is promoted into the central Git™ repository only if it meets safety and security requirements.

Polyspace as You Code: static analysis integrated in your environment



**Polyspace as You Code is a feature of Polyspace Access*

First opportunity to fix bugs…



Also supported:

and custom integrations

…on demand.

…before committing.

…before running tests.

…while you remember the code.

…when it's easiest.

…when it's least expensive.

\+ Help develop good habits

Find Bugs and Enforce Coding Standards

Defect Types

- Numerical
- Data Flow
- Good Practice
- Tainted Data
- Concurrency
- Performance
- Security
- Static Memory
- Resource Mgmt.
- Cryptography
- Dynamic Memory
- Programming

Coding Standards

- MISRA C:2004
- MISRA C++:2008
- Naming Rules
- MISRA C:2012
- AUTOSAR C++-14
- JSF AV C++
- CERT C
- CERT C++
- ISO/IEC TS 17961

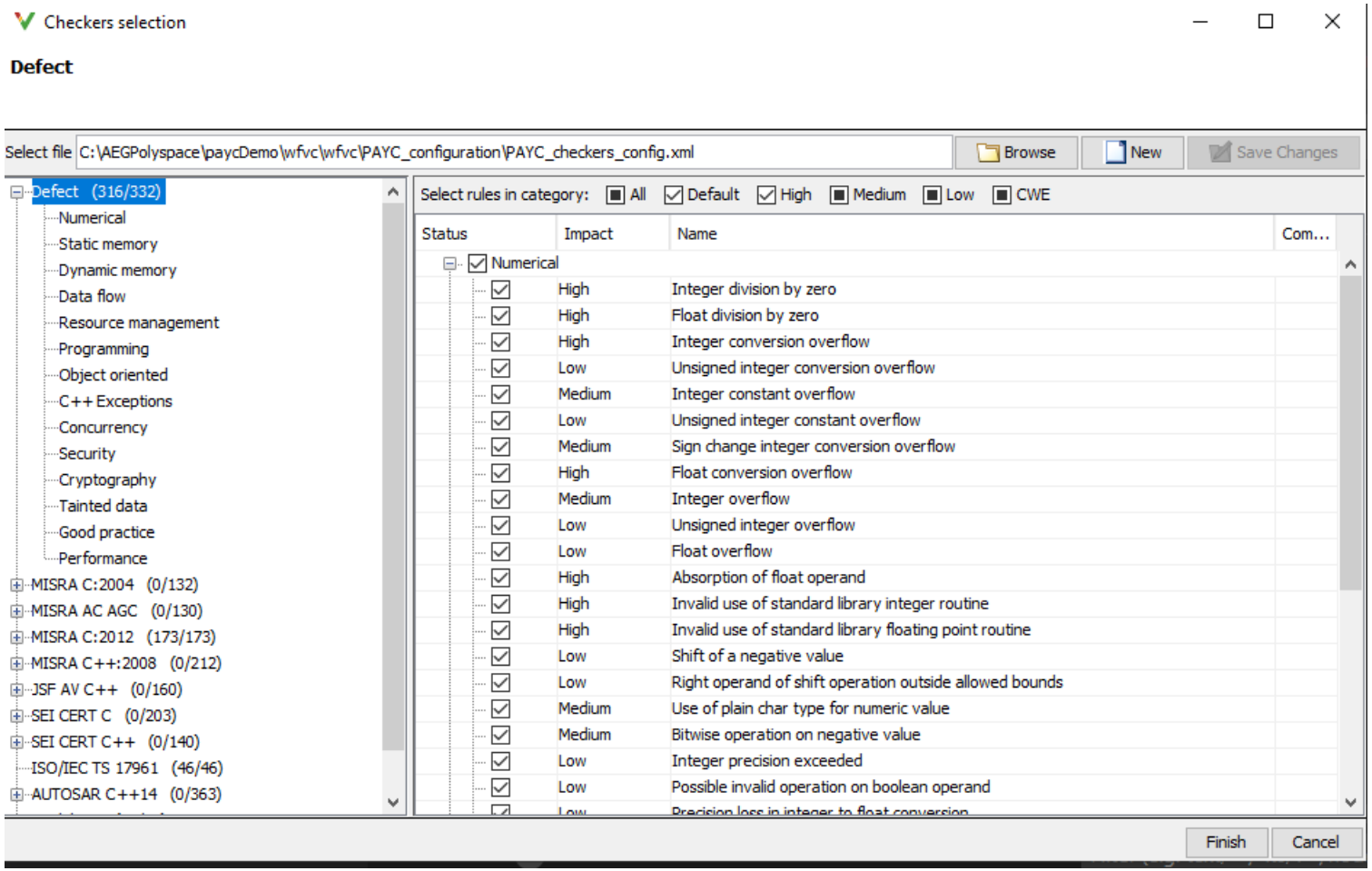
Guidelines checks for software metrics

Guidelines

- Complexity
- Recursions
- Language Scope
- Function Coupling
- HIS
- Paths, Inputs, Calls
- Project
- File
- Function

Configuration

- File based configuration can be centralized and shared across teams
- Can also import from Polyspace Desktop or use options text files



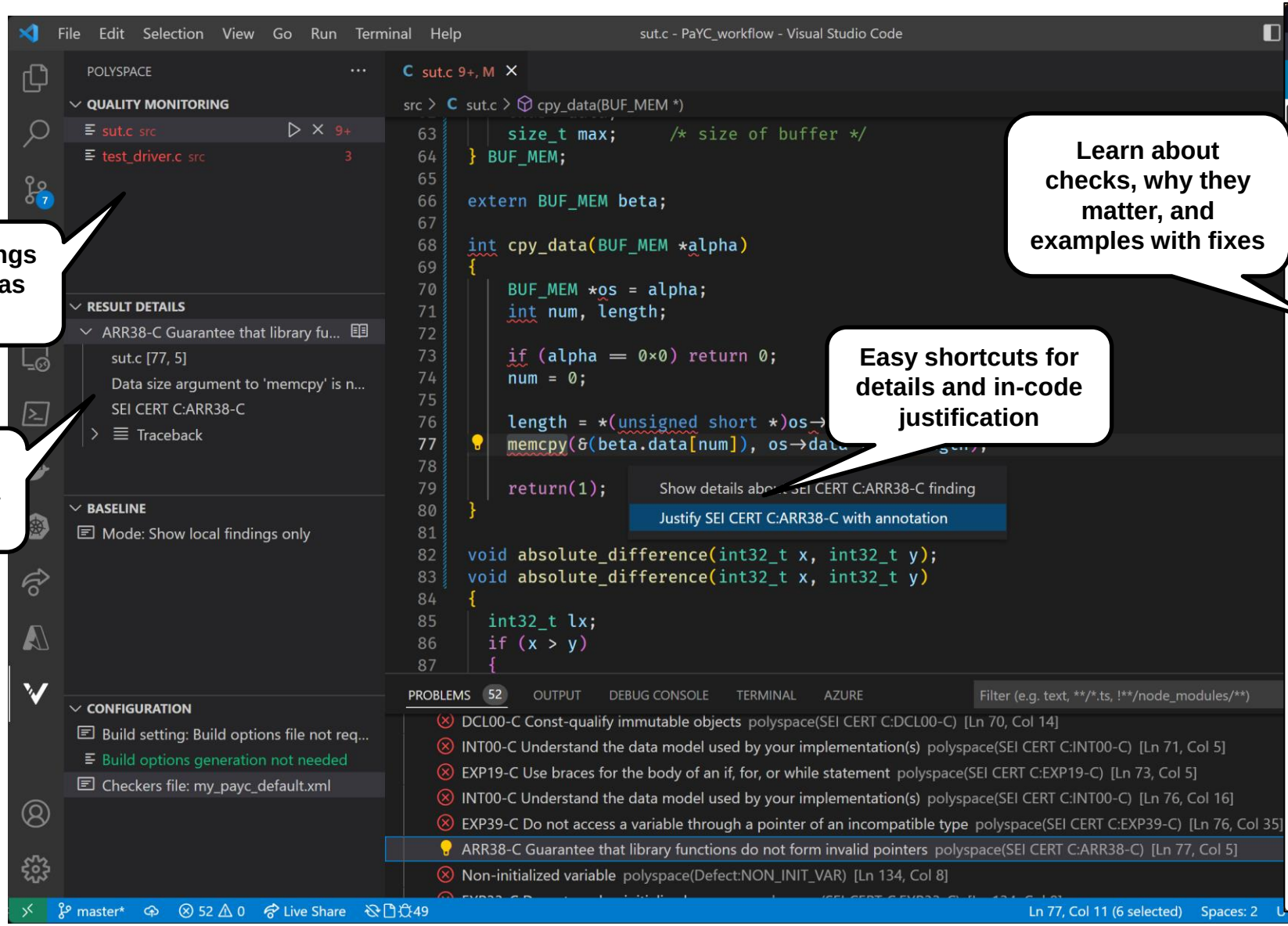
Polyspace as You Code for all the C[++] code you write

Monitor findings for key files as you code

See finding details with a traceback

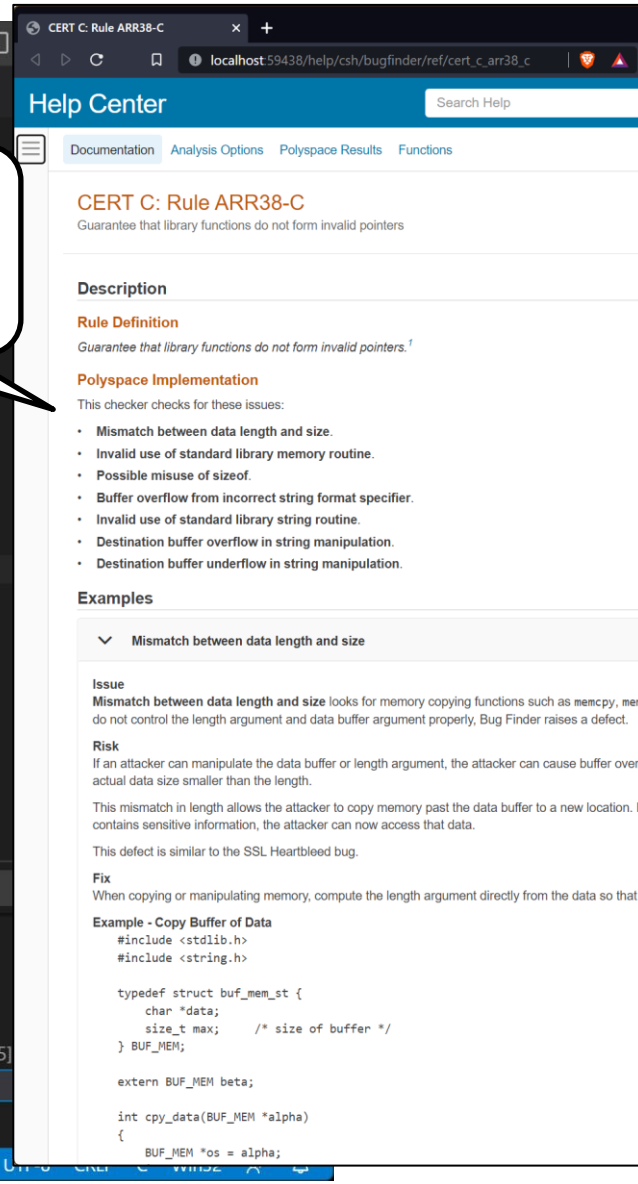
Also supported:

and custom integrations

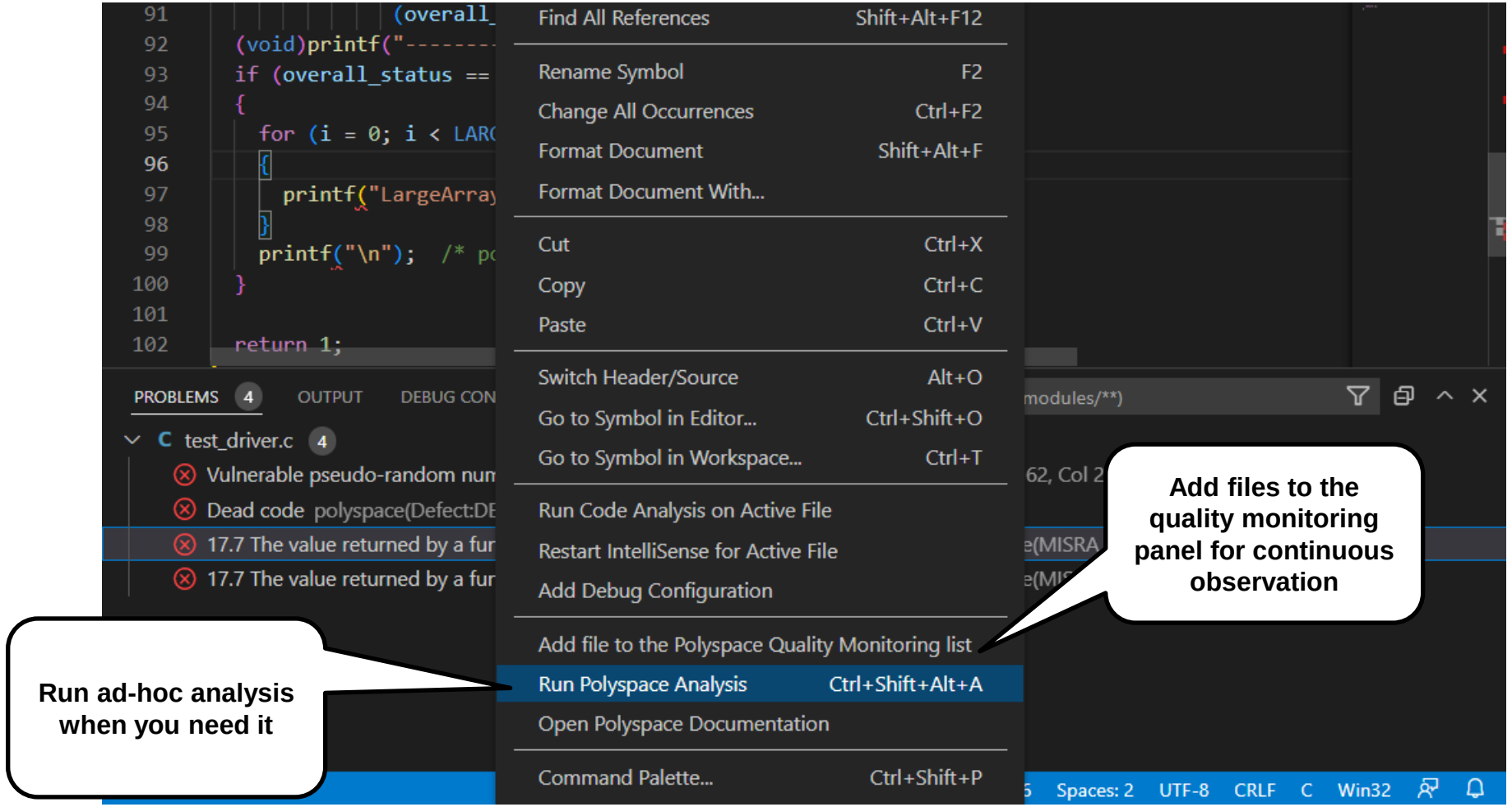


Learn about checks, why they matter, and examples with fixes

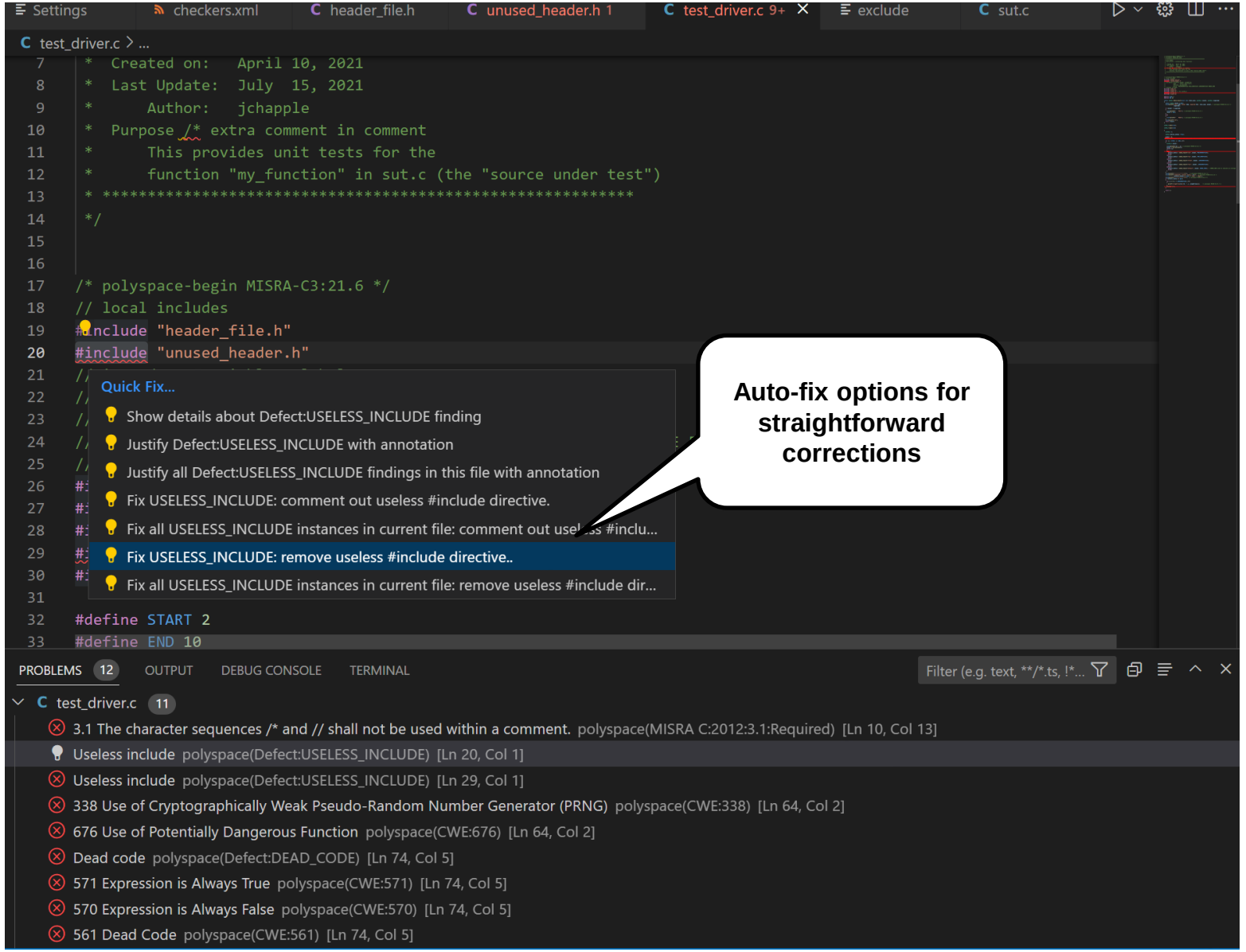
Easy shortcuts for details and in-code justification



Fast local analysis on save and on-demand



Productivity enhancing auto-fix feature



Demo

Come by the booth or visit our website

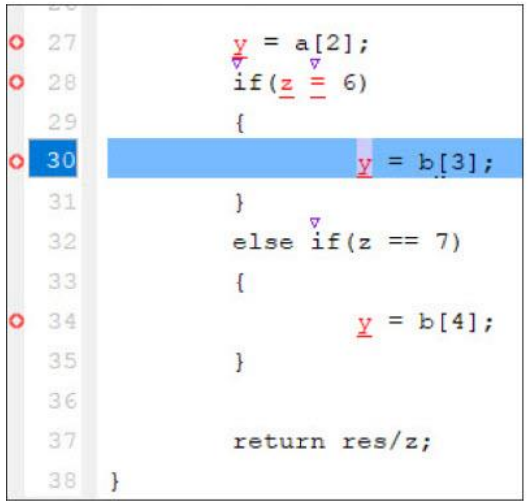
Get Polyspace as You Code

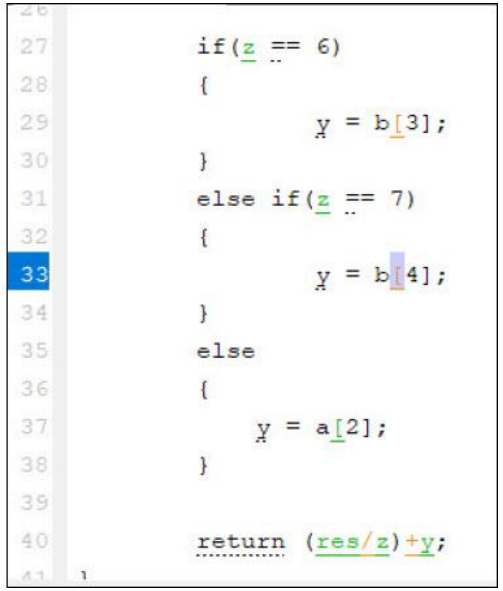
- https://www.mathworks.com/downloads/csd/Polyspace/23a/PaYC_Installer_win64.zip
- https://www.mathworks.com/downloads/csd/Polyspace/23a/PaYC_Installer_glnxa64.zip

Training: Polyspace for C/C++ Code Verification

After this 2-day course you will be able to:

- Create a verification project
- Review verification results
- Emulate target execution environments
- Handle missing functions and data
- Manage unproven code
- Apply MISRA-C® rules
- Report verification results via Polyspace Access





Upcoming offerings: May 2-3, June 22-23