C and C++ DevOps with GitLab, Visual Studio Code, and Polyspace

Matt Rhodes

Presented 4 June 2025

Attendee Instructions

All participants are muted

Please ask questions in the Chat

Slides to be shared after the event

Agenda (45min)

- Polyspace Capabilities
- Shift-Left
- Automation

Polyspace Capabilities

Top 3 Values of Polyspace

1. Proof of Robustness	← More Certainty
2. Shift Left, aka “Best place to fix” workflow	← More Efficiency
3. Adaptable tooling and interfaces	← More Flexibility

Less Sacrifice on Quality

Top 3 Values of Polyspace

1. Proof of Robustness	← More Certainty
2. Shift Left, aka “Best place to fix” workflow	← More Efficiency
3. Adaptable tooling and interfaces	← More Flexibility

TODAY

Polyspace Static Analysis Objectives

Safety	Security
Standards: • DO-178 (aero) • ISO 26262 (auto) • IEC 61508 (industrial) • EN 50128 (rail *control*) • EN 50657 (rail *roll. stock*) • IEC 62304 (med) • ISO 25119 (agr) • MISRA • AUTOSAR	**Broad Coverage:** • CERT, CWE, ISO 17961, ISO 21434 (auto) • MISRA-C:2023, MISRA-C++:2023 • Custom: DISA STIG, HMC guidelines, etc. • Security, Cryptography, Tainted data • Proof of absence of runtime vulnerabilities

Proof of Robustness

Code Proving via Abstract Interpretation

- Prove absence of critical runtime errors (or find even the slightest vulnerability)
- Exhaustive: all possible inputs, control flows, data flows (no instrumentation, execution, test cases)
- Sound: no false negatives

Quality

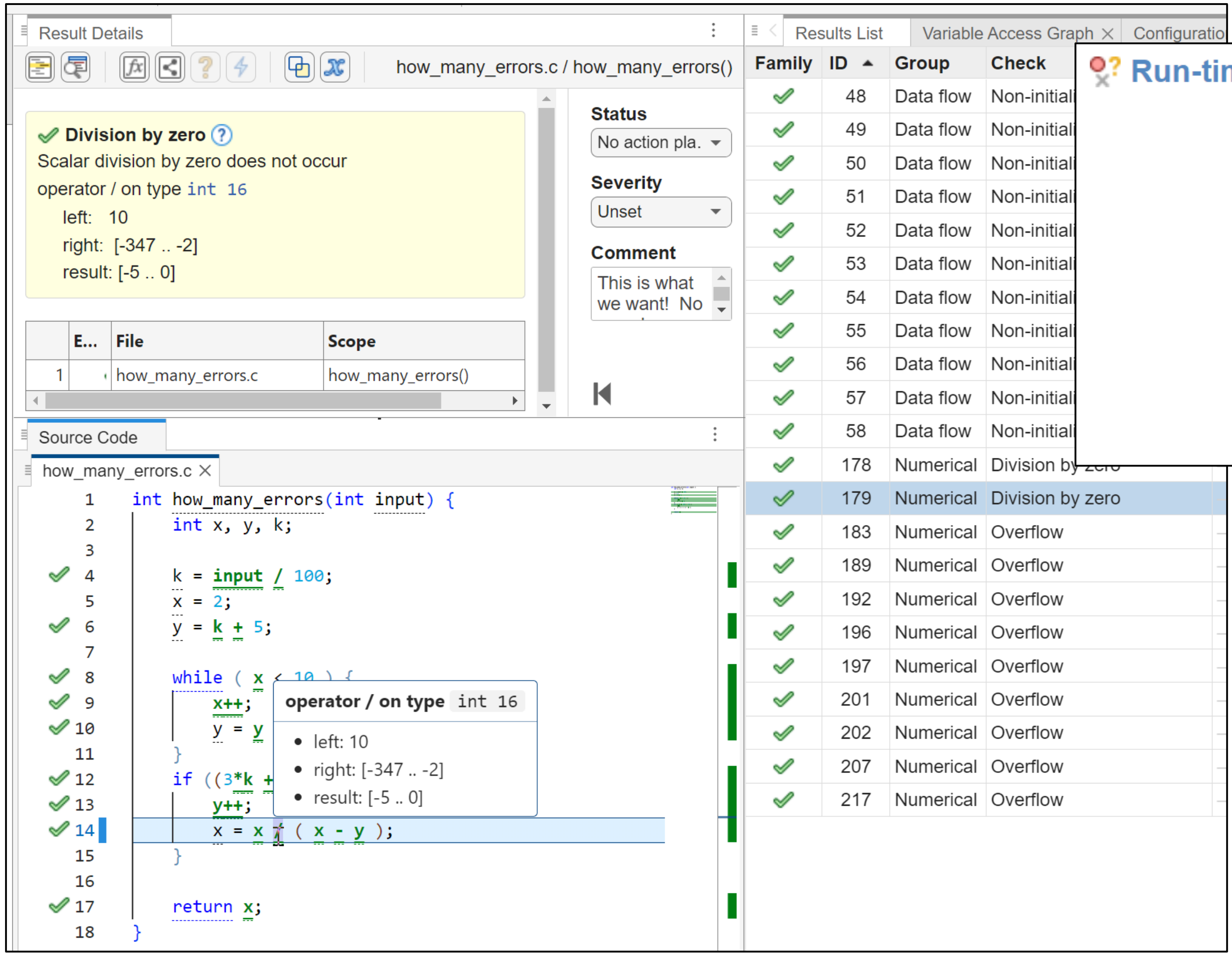
"Traditional" Static Analysis

- Coding Standards
- Find Probable Bugs, Defects
- Code Metrics

Plus…

- Formal Method: Runtime Behavior, Debugger-like view
- Review Scopes / Software Quality Objectives
- Simulink Integration: trace issues in generated code back to model

Polyspace Code Prover



Result Details

how_many_errors.c / how_many_errors()

✓ **Division by zero**
Scalar division by zero does not occur
operator / on type `int 16`
left: 10
right: [-347 .. -2]
result: [-5 .. 0]

Status
No action pla.

Severity
Unset

Comment
This is what we want! No

E...	File	Scope
1	how_many_errors.c	how_many_errors()

Source Code

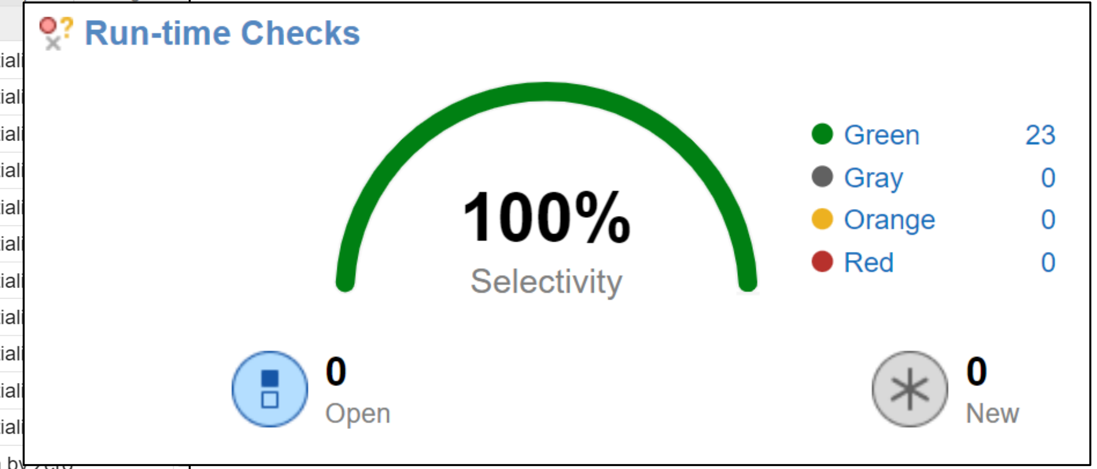
how_many_errors.c

```
int how_many_errors(int input) {
    int x, y, k;

    k = input / 100;
    x = 2;
    y = k + 5;

    while ( x < 10 ) {
        x++;
        y = y
    }
    if ((3*k +
        y++;
        x = x / ( x - y );
    }

    return x;
}
```

operator / on type `int 16`
- left: 10
- right: [-347 .. -2]
- result: [-5 .. 0]

Results List | Variable Access Graph | Configuratio

Family	ID	Group	Check
✓	48	Data flow	Non-initiali
✓	49	Data flow	Non-initiali
✓	50	Data flow	Non-initiali
✓	51	Data flow	Non-initiali
✓	52	Data flow	Non-initiali
✓	53	Data flow	Non-initiali
✓	54	Data flow	Non-initiali
✓	55	Data flow	Non-initiali
✓	56	Data flow	Non-initiali
✓	57	Data flow	Non-initiali
✓	58	Data flow	Non-initiali
✓	178	Numerical	Division by zero
✓	179	Numerical	Division by zero
✓	183	Numerical	Overflow
✓	189	Numerical	Overflow
✓	192	Numerical	Overflow
✓	196	Numerical	Overflow
✓	197	Numerical	Overflow
✓	201	Numerical	Overflow
✓	202	Numerical	Overflow
✓	207	Numerical	Overflow
✓	217	Numerical	Overflow



Run-time Checks

100%
Selectivity

● Green	23
● Gray	0
● Orange	0
● Red	0

0 Open

0 New

Proof of robustness against unknown vulnerabilities

Polyspace Tools

Bug Finder

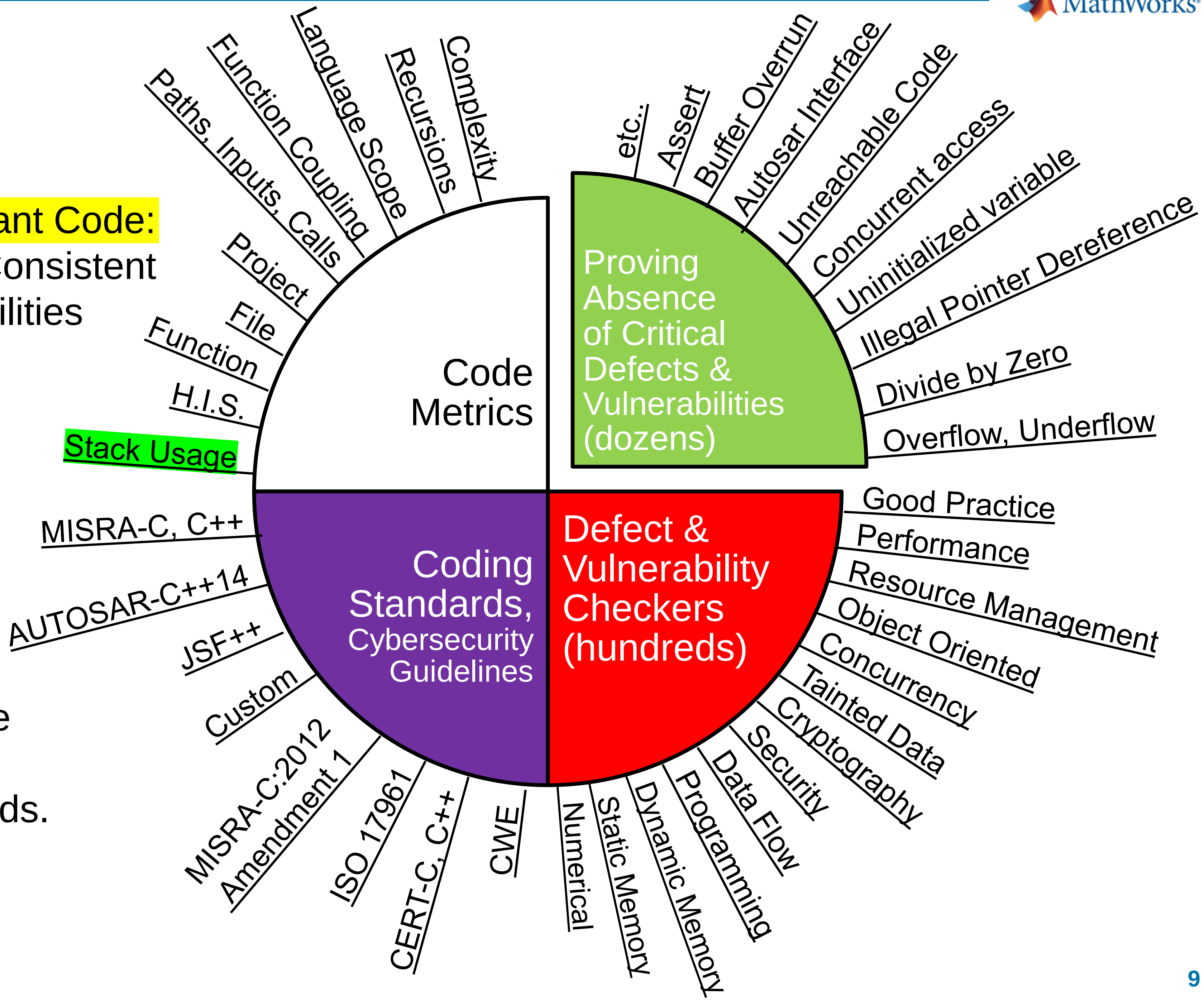
→High Quality, Secure, Compliant Code:

- Measurable, Maintainable, Consistent
- Very few defects or vulnerabilities
- Credits for functional safety, cybersecurity standards.

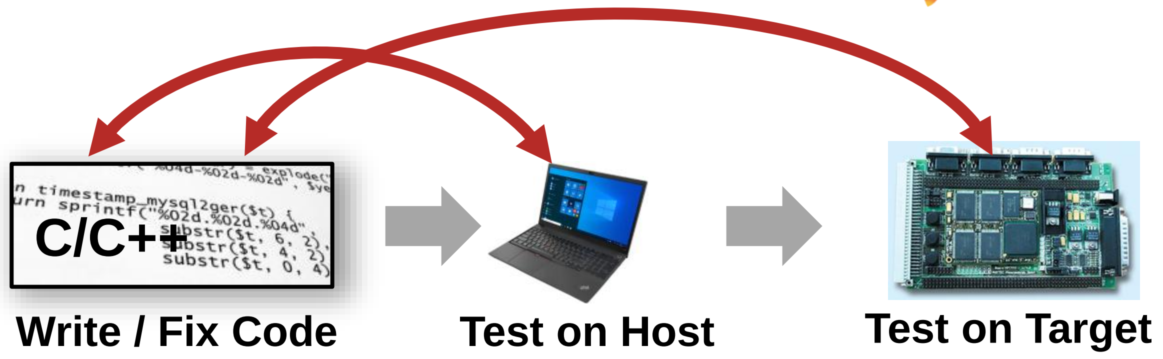
Code Prover

→Zero-Defect Code:

- Robust, Safe, Secure
- Proven free of critical runtime defects and vulnerabilities
- Additional credits for standards.



Polyspace Test Major Capabilities



xUnit or GUI workflows

Mock/stub, link requirements

Automatic test gen

Authoring

Build tests

Manage test execution

Run on host or target

Execution

Pass / Fail

Coverage analysis

Profile exec, memory

Review

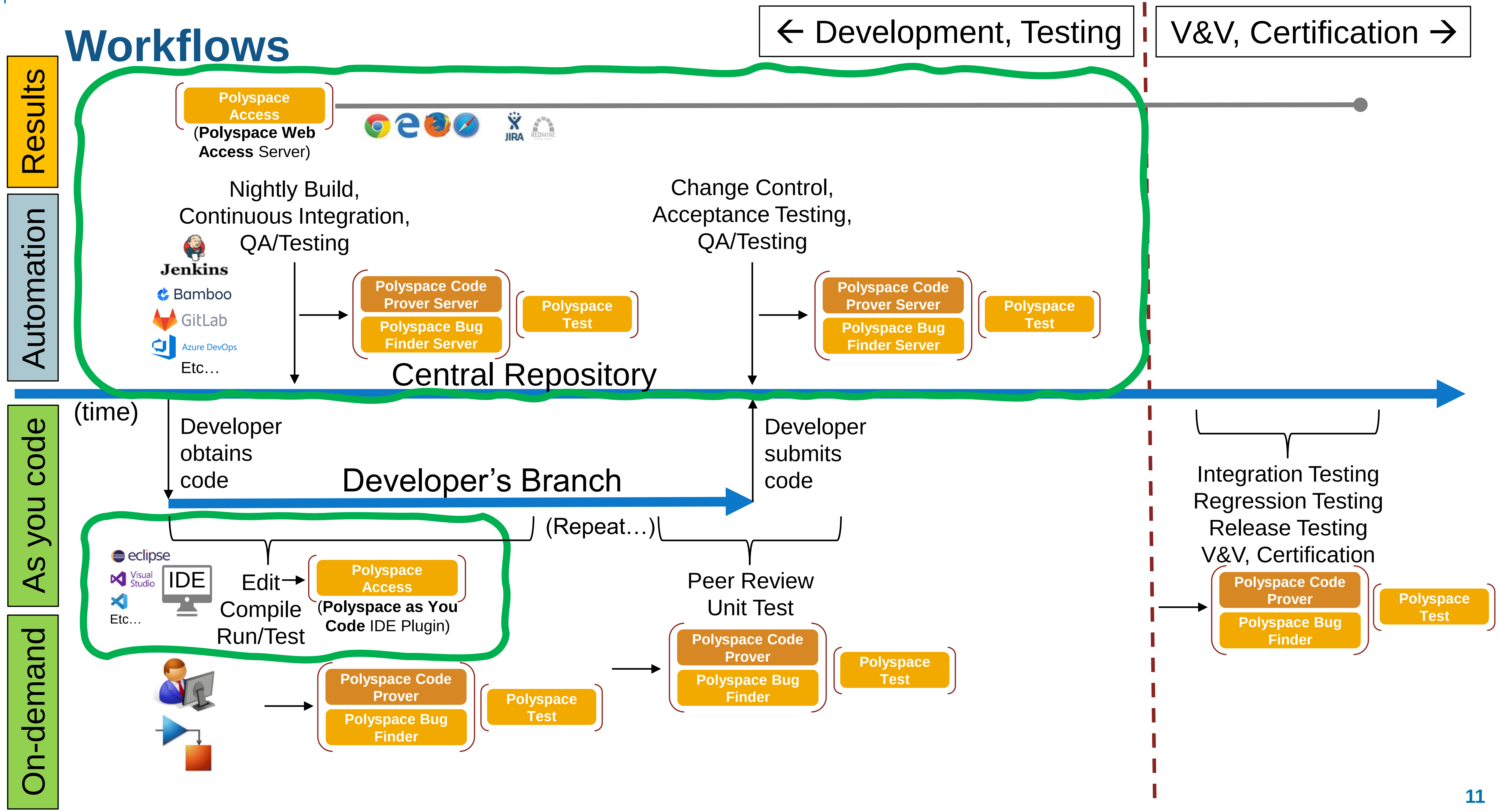
- IDE or Command Line
- Desktop GUI

Developers and Testers

- Continuous Integration
- Results via browser

Team Collaboration

Workflows


← Development, Testing
V&V, Certification →
Results
Automation
As you code
On-demand
Polyspace Access
(Polyspace Web Access Server)
JIRA
REDMINE
Nightly Build, Continuous Integration, QA/Testing
Jenkins
Bamboo
GitLab
Azure DevOps
Etc…
Polyspace Code Prover Server
Polyspace Bug Finder Server
Polyspace Test
Change Control, Acceptance Testing, QA/Testing
Polyspace Code Prover Server
Polyspace Bug Finder Server
Polyspace Test
Central Repository
(time)
Developer obtains code
Developer's Branch
Developer submits code
(Repeat…)
eclipse
Visual Studio
IDE
Etc…
Edit
Compile
Run/Test
Polyspace Access
(Polyspace as You Code IDE Plugin)
Peer Review
Unit Test
Polyspace Code Prover
Polyspace Bug Finder
Polyspace Test
Polyspace Code Prover
Polyspace Bug Finder
Polyspace Test
Integration Testing
Regression Testing
Release Testing
V&V, Certification
Polyspace Code Prover
Polyspace Bug Finder
Polyspace Test

Developer Efficiency → Shift-Left

Comprehensive static analysis for increased efficiency

Catch and fix bugs while you code

Automate with CI Workflows

Collaborate with Team Members

What are some of the actionable steps?

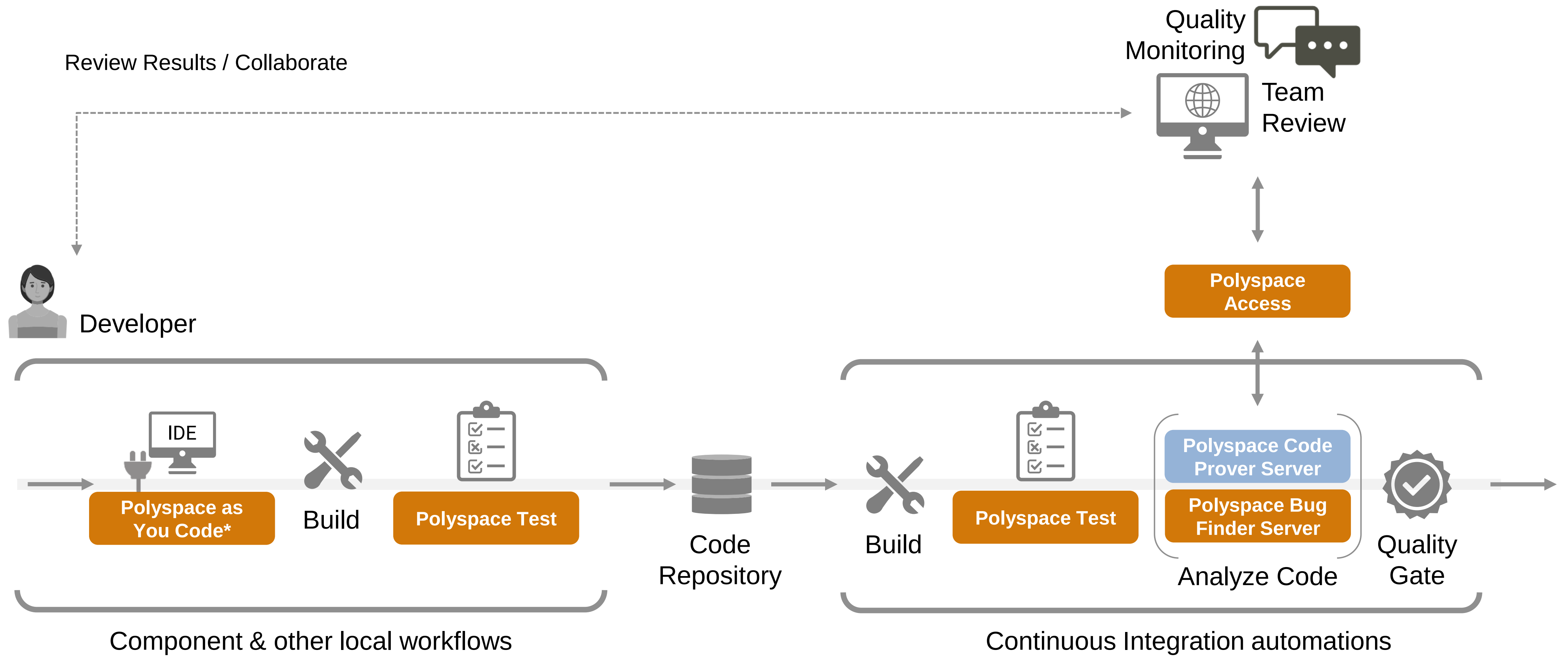
<< Shift Left <<

Catch and fix bugs while you code

Automate + use CI Workflows

Collaborate with Team Members

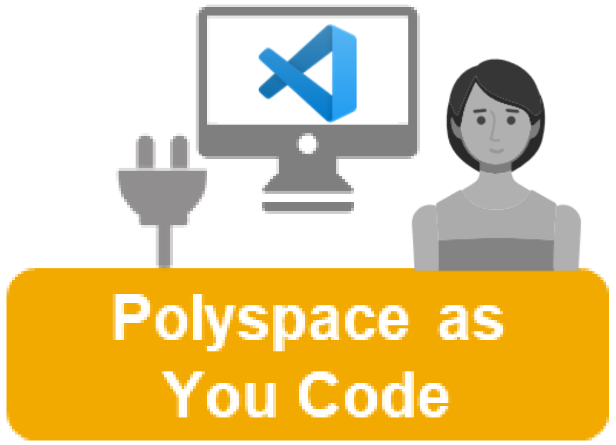
Comprehensive static analysis for sanity in your Dev[Sec]Ops



<< SHIFT LEFT <<

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.

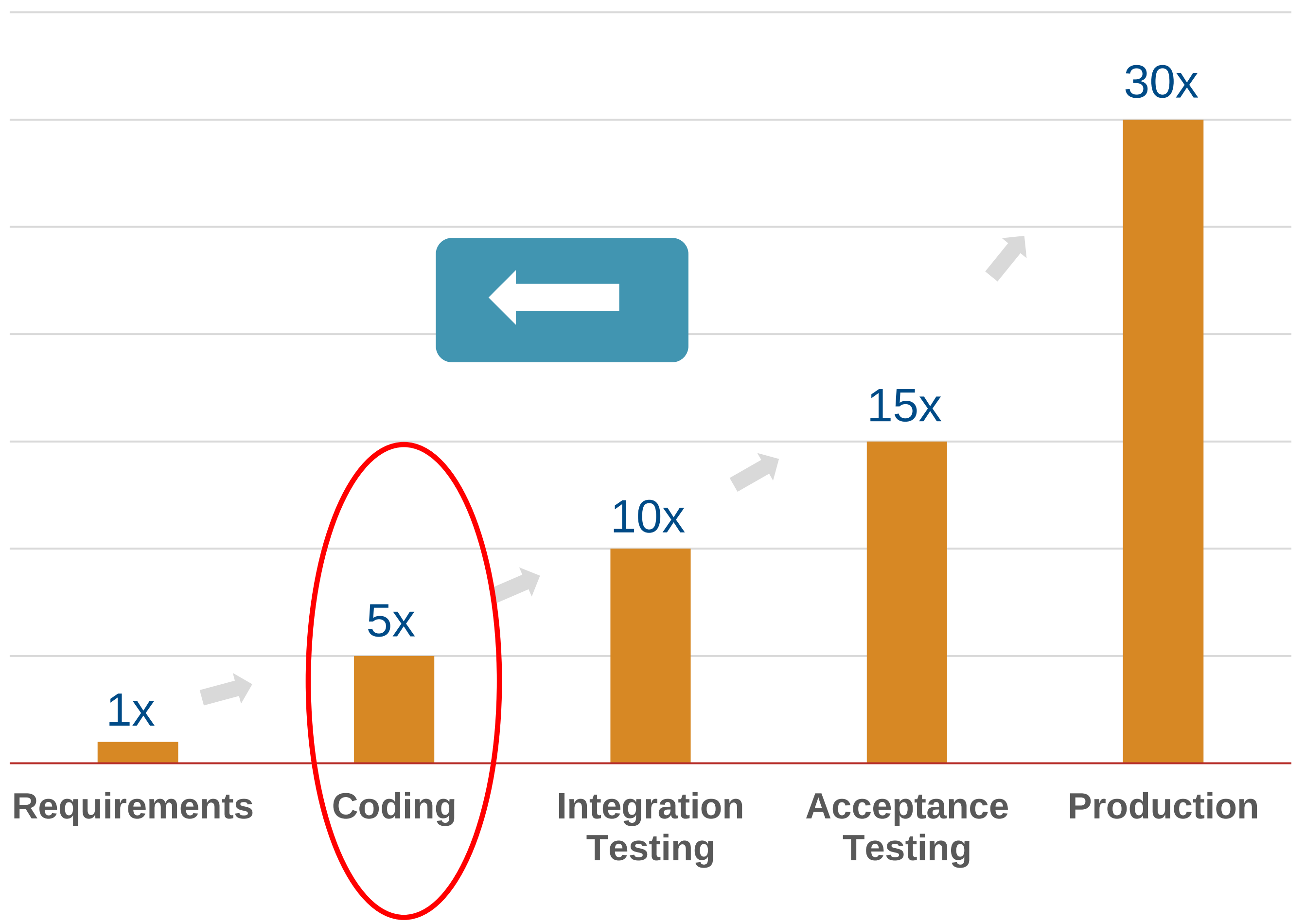
…while you remember the code.

…when it's easiest.

…when it's least expensive.

+ Help develop good habits

Reduce cost with earliest verification



Source: National Institute of Standards and Technology (NIST)

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 Convention
❖ MISRA C:2012	❖ AUTOSAR C++-14	❖ JSF AV C++
❖ CERT C	❖ CERT C++	❖ ISO/IEC TS 17961

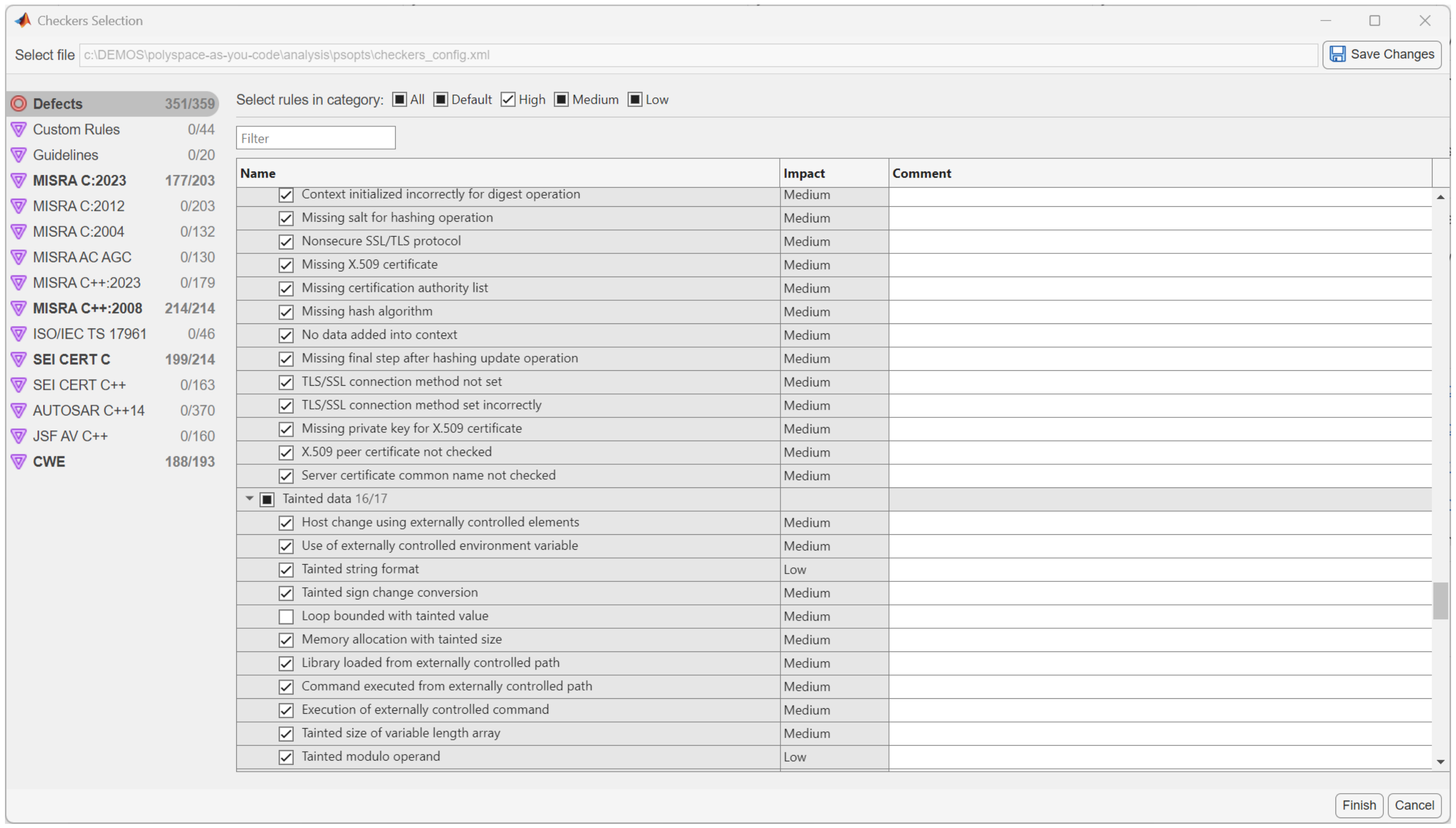
Guidelines checks for software metrics

Guidelines

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

Configuration

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



Checkers Selection

Select file: c:\DEMOS\polyspace-as-you-code\analysis\psopts\checkers_config.xml — Save Changes

Defects	**351/359**
Custom Rules	0/44
Guidelines	0/20
MISRA C:2023	**177/203**
MISRA C:2012	0/203
MISRA C:2004	0/132
MISRA AC AGC	0/130
MISRA C++:2023	0/179
MISRA C++:2008	**214/214**
ISO/IEC TS 17961	0/46
SEI CERT C	**199/214**
SEI CERT C++	0/163
AUTOSAR C++14	0/370
JSF AV C++	0/160
CWE	**188/193**

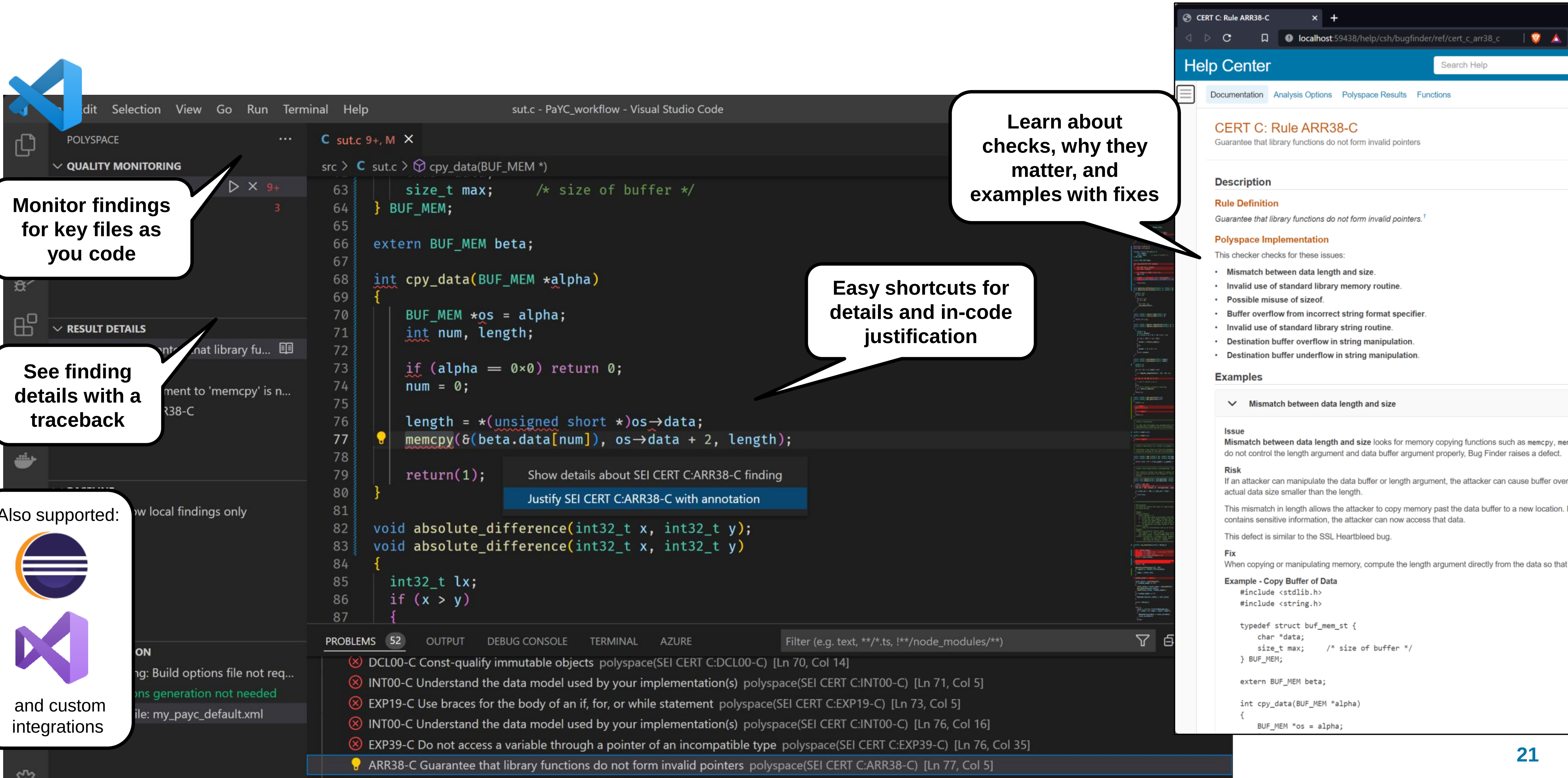
Select rules in category: ■ All ■ Default ✓ High ■ Medium ■ Low

Filter

Name	Impact	Comment
✓ Context initialized incorrectly for digest operation	Medium	
✓ Missing salt for hashing operation	Medium	
✓ Nonsecure SSL/TLS protocol	Medium	
✓ Missing X.509 certificate	Medium	
✓ Missing certification authority list	Medium	
✓ Missing hash algorithm	Medium	
✓ No data added into context	Medium	
✓ Missing final step after hashing update operation	Medium	
✓ TLS/SSL connection method not set	Medium	
✓ TLS/SSL connection method set incorrectly	Medium	
✓ Missing private key for X.509 certificate	Medium	
✓ X.509 peer certificate not checked	Medium	
✓ Server certificate common name not checked	Medium	
■ Tainted data 16/17		
✓ Host change using externally controlled elements	Medium	
✓ Use of externally controlled environment variable	Medium	
✓ Tainted string format	Low	
✓ Tainted sign change conversion	Medium	
☐ Loop bounded with tainted value	Medium	
✓ Memory allocation with tainted size	Medium	
✓ Library loaded from externally controlled path	Medium	
✓ Command executed from externally controlled path	Medium	
✓ Execution of externally controlled command	Medium	
✓ Tainted size of variable length array	Medium	
✓ Tainted modulo operand	Low	

Finish Cancel

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


dit Selection View Go Run Terminal Help
sut.c - PaYC_workflow - Visual Studio Code
POLYSPACE
QUALITY MONITORING
RESULT DETAILS
C sut.c 9+, M
src > C sut.c > cpy_data(BUF_MEM *)
Show details about SEI CERT C:ARR38-C finding
Justify SEI CERT C:ARR38-C with annotation
PROBLEMS 52 OUTPUT DEBUG CONSOLE TERMINAL AZURE
Filter (e.g. text, **/*.ts, !**/node_modules/**)
CERT C: Rule ARR38-C
localhost:59438/help/csh/bugfinder/ref/cert_c_arr38_c
Help Center
Search Help
Documentation Analysis Options Polyspace Results Functions

```
    size_t max;      /* size of buffer */
} BUF_MEM;

extern BUF_MEM beta;

int cpy_data(BUF_MEM *alpha)
{
    BUF_MEM *os = alpha;
    int num, length;

    if (alpha == 0x0) return 0;
    num = 0;

    length = *(unsigned short *)os->data;
    memcpy(&(beta.data[num]), os->data + 2, length);

    return(1);
}

void absolute_difference(int32_t x, int32_t y);
void absolute_difference(int32_t x, int32_t y)
{
  int32_t lx;
  if (x > y)
  {
```

```
nat library fu...
ment to 'memcpy' is n...
38-C
w local findings only
ON
g: Build options file not req...
ns generation not needed
ile: my_payc_default.xml
```

```
DCL00-C Const-qualify immutable objects polyspace(SEI CERT C:DCL00-C) [Ln 70, Col 14]
INT00-C Understand the data model used by your implementation(s) polyspace(SEI CERT C:INT00-C) [Ln 71, Col 5]
EXP19-C Use braces for the body of an if, for, or while statement polyspace(SEI CERT C:EXP19-C) [Ln 73, Col 5]
INT00-C Understand the data model used by your implementation(s) polyspace(SEI CERT C:INT00-C) [Ln 76, Col 16]
EXP39-C Do not access a variable through a pointer of an incompatible type polyspace(SEI CERT C:EXP39-C) [Ln 76, Col 35]
ARR38-C Guarantee that library functions do not form invalid pointers polyspace(SEI CERT C:ARR38-C) [Ln 77, Col 5]
```

CERT C: Rule ARR38-C
Guarantee that library functions do not form invalid pointers

Description

Rule Definition

Guarantee that library functions do not form invalid pointers.[1]

Polyspace Implementation

This checker checks for these issues:

- **Mismatch between data length and size.**
- **Invalid use of standard library memory routine.**
- **Possible misuse of sizeof.**
- **Buffer overflow from incorrect string format specifier.**
- **Invalid use of standard library string routine.**
- **Destination buffer overflow in string manipulation.**
- **Destination buffer underflow in string manipulation.**

Examples

Mismatch between data length and size

Issue
Mismatch between data length and size looks for memory copying functions such as `memcpy`, me
do not control the length argument and data buffer argument properly, Bug Finder raises a defect.

Risk
If an attacker can manipulate the data buffer or length argument, the attacker can cause buffer over
actual data size smaller than the length.

This mismatch in length allows the attacker to copy memory past the data buffer to a new location.
contains sensitive information, the attacker can now access that data.

This defect is similar to the SSL Heartbleed bug.

Fix
When copying or manipulating memory, compute the length argument directly from the data so that

Example - Copy Buffer of Data

```
#include <stdlib.h>
#include <string.h>

typedef struct buf_mem_st {
    char *data;
    size_t max;     /* size of buffer */
} BUF_MEM;

extern BUF_MEM beta;

int cpy_data(BUF_MEM *alpha)
{
    BUF_MEM *os = alpha;
```

Monitor findings for key files as you code

See finding details with a traceback

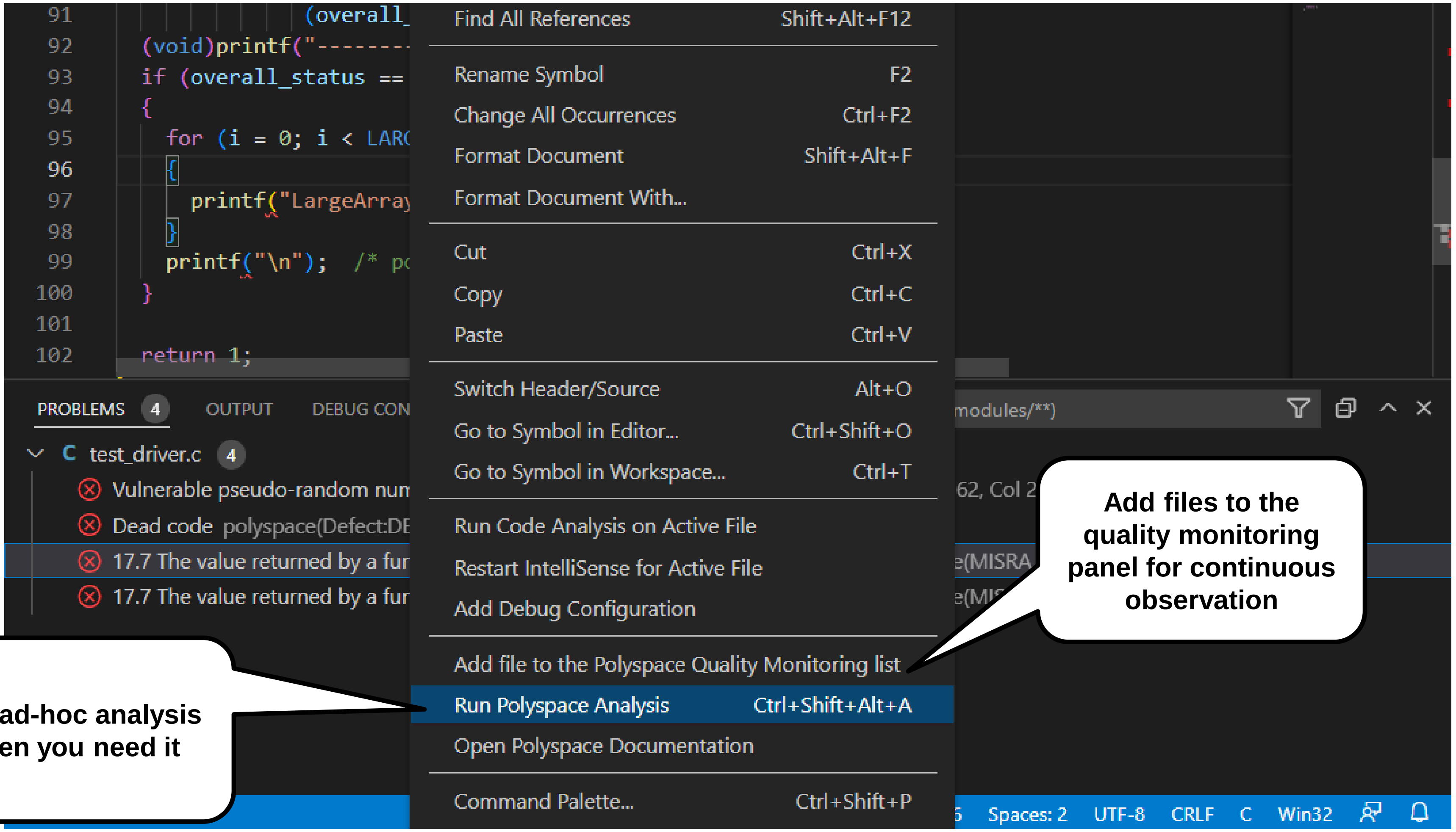
Learn about checks, why they matter, and examples with fixes

Easy shortcuts for details and in-code justification

Also supported:

and custom integrations

Fast local analysis on save and on-demand



```
                  (overall_
(void)printf("-------
if (overall_status ==
{
  for (i = 0; i < LAR(
  {
    printf("LargeArray
  }
  printf("\n");  /* po
}

return 1;
```

PROBLEMS 4 OUTPUT DEBUG CON

test_driver.c 4

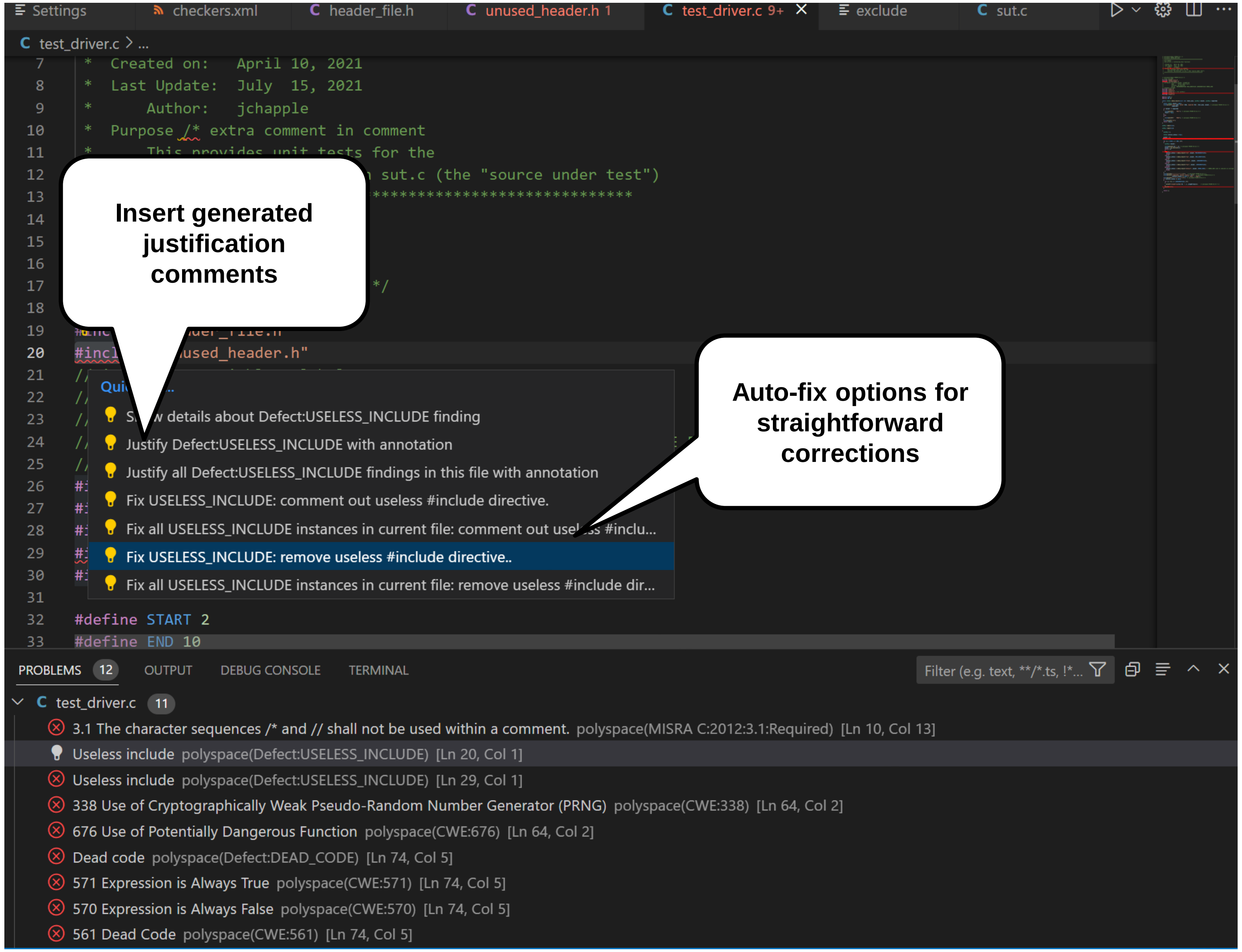
- Vulnerable pseudo-random num
- Dead code polyspace(Defect:DE
- 17.7 The value returned by a fur
- 17.7 The value returned by a fur

Find All References Shift+Alt+F12
Rename Symbol F2
Change All Occurrences Ctrl+F2
Format Document Shift+Alt+F
Format Document With...
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Switch Header/Source Alt+O
Go to Symbol in Editor... Ctrl+Shift+O
Go to Symbol in Workspace... Ctrl+T
Run Code Analysis on Active File
Restart IntelliSense for Active File
Add Debug Configuration
Add file to the Polyspace Quality Monitoring list
Run Polyspace Analysis Ctrl+Shift+Alt+A
Open Polyspace Documentation
Command Palette... Ctrl+Shift+P
modules/**)
62, Col 2
e(MISRA
e(MIS
Spaces: 2 UTF-8 CRLF C Win32

Add files to the quality monitoring panel for continuous observation

Run ad-hoc analysis when you need it

Auto Fix



```
 7  *  Created on:   April 10, 2021
 8  *  Last Update:  July  15, 2021
 9  *      Author:   jchapple
10  *  Purpose /* extra comment in comment
11  *      This provides unit tests for the
12                                  sut.c (the "source under test")
13                                  ****************************
14
15
16
17                                  */
18
19  #inc          der_file.h
20  #incl         used_header.h"
21  //
22  //  Qui
23  //
24  //
25  //
26  #
27  #
28  #
29  #
30  #
31
32  #define START 2
33  #define END 10
```

Quick Fix...
- Show details about Defect:USELESS_INCLUDE finding
- Justify Defect:USELESS_INCLUDE with annotation
- Justify all Defect:USELESS_INCLUDE findings in this file with annotation
- Fix USELESS_INCLUDE: comment out useless #include directive.
- Fix all USELESS_INCLUDE instances in current file: comment out useless #inclu...
- Fix USELESS_INCLUDE: remove useless #include directive..
- Fix all USELESS_INCLUDE instances in current file: remove useless #include dir...

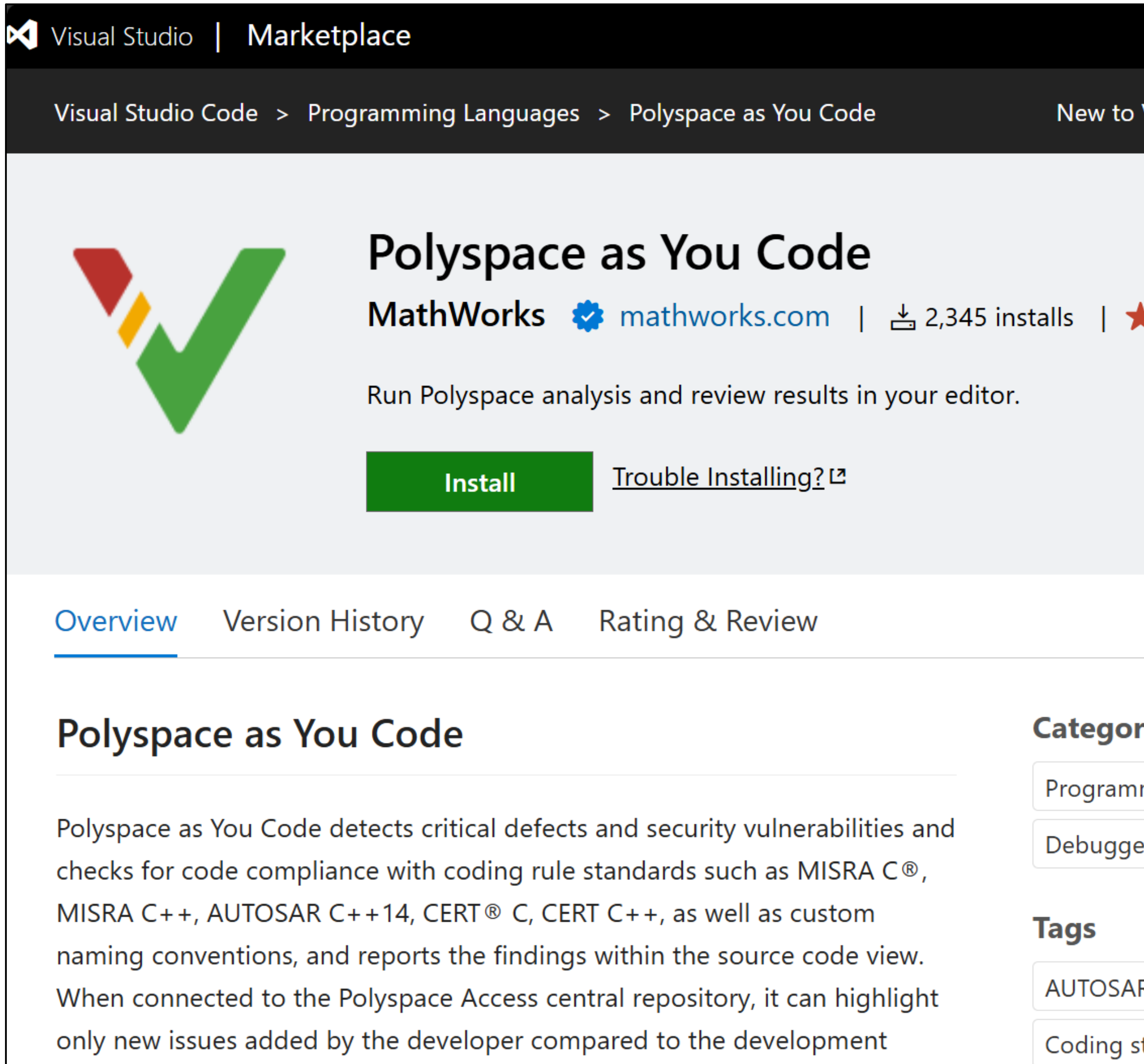
PROBLEMS 12   OUTPUT   DEBUG CONSOLE   TERMINAL   Filter (e.g. text, **/*.ts, !*...

test_driver.c 11
- 3.1 The character sequences /* and // shall not be used within a comment. polyspace(MISRA C:2012:3.1:Required) [Ln 10, Col 13]
- Useless include polyspace(Defect:USELESS_INCLUDE) [Ln 20, Col 1]
- Useless include polyspace(Defect:USELESS_INCLUDE) [Ln 29, Col 1]
- 338 Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG) polyspace(CWE:338) [Ln 64, Col 2]
- 676 Use of Potentially Dangerous Function polyspace(CWE:676) [Ln 64, Col 2]
- Dead code polyspace(Defect:DEAD_CODE) [Ln 74, Col 5]
- 571 Expression is Always True polyspace(CWE:571) [Ln 74, Col 5]
- 570 Expression is Always False polyspace(CWE:570) [Ln 74, Col 5]
- 561 Dead Code polyspace(CWE:561) [Ln 74, Col 5]

Available in the VS Code Marketplace

- Polyspace Access license required
  - Free trial available

- Extension requires that you install the Polyspace as You Code engine
  - links available on the marketplace page
    - Windows
    - Linux
    - macOS



Polyspace as You Code

MathWorks mathworks.com | 2,345 installs |

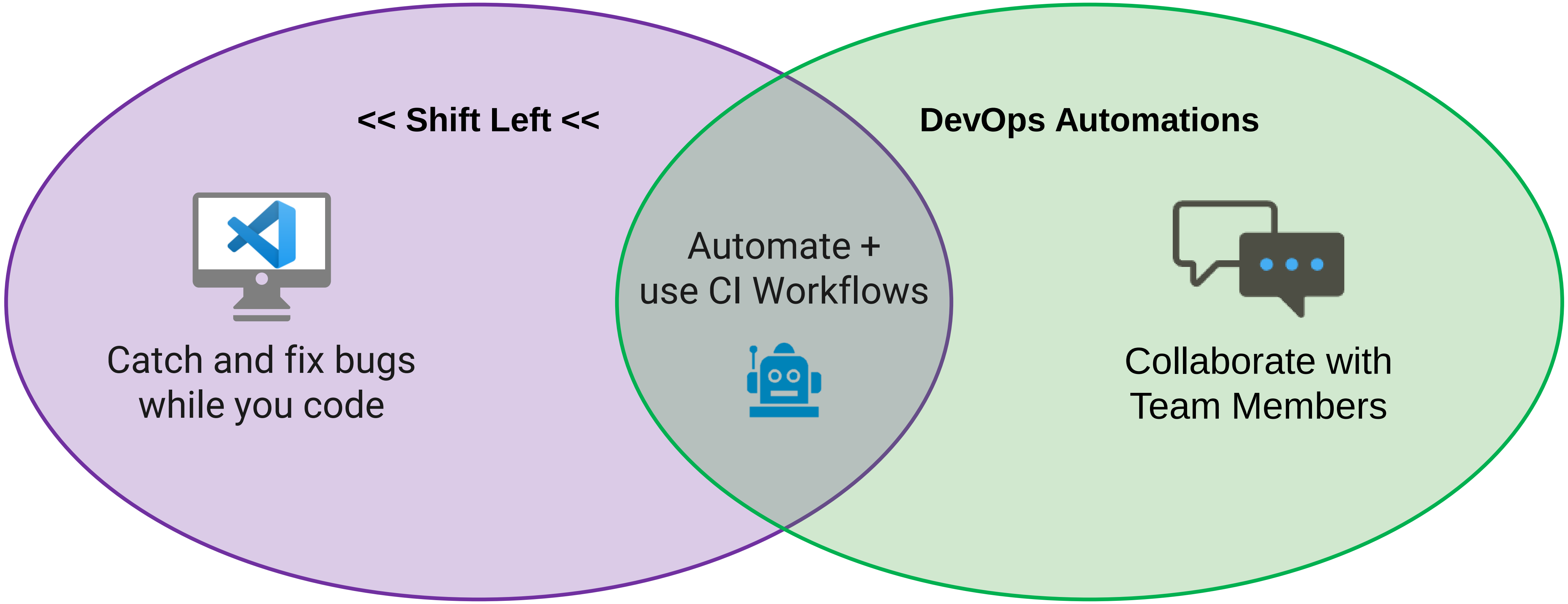
Run Polyspace analysis and review results in your editor.

Polyspace as You Code

Polyspace as You Code detects critical defects and security vulnerabilities and checks for code compliance with coding rule standards such as MISRA C®, MISRA C++, AUTOSAR C++14, CERT® C, CERT C++, as well as custom naming conventions, and reports the findings within the source code view. When connected to the Polyspace Access central repository, it can highlight only new issues added by the developer compared to the development

Developer Efficiency → Automation

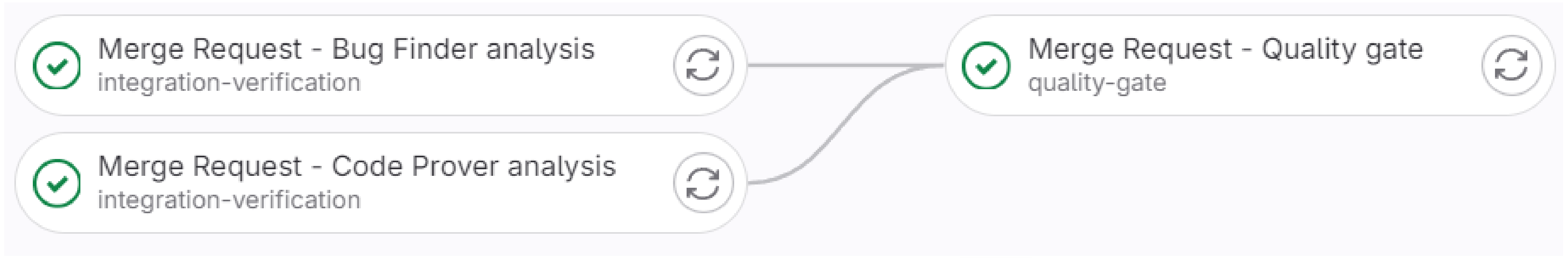
What are some of the actionable steps?



<< Shift Left <<
DevOps Automations
Catch and fix bugs while you code
Automate + use CI Workflows
Collaborate with Team Members

Leverage [a | your] CI system

- Automation of quality gates
  - Unit level
  - Integration level
- Coordinate
  - What-if scenarios with feature branches
  - Merge/Pull requests for aligning work


Merge Request - Bug Finder analysis
integration-verification

Merge Request - Code Prover analysis
integration-verification

Merge Request - Quality gate
quality-gate

Jenkins

Bamboo

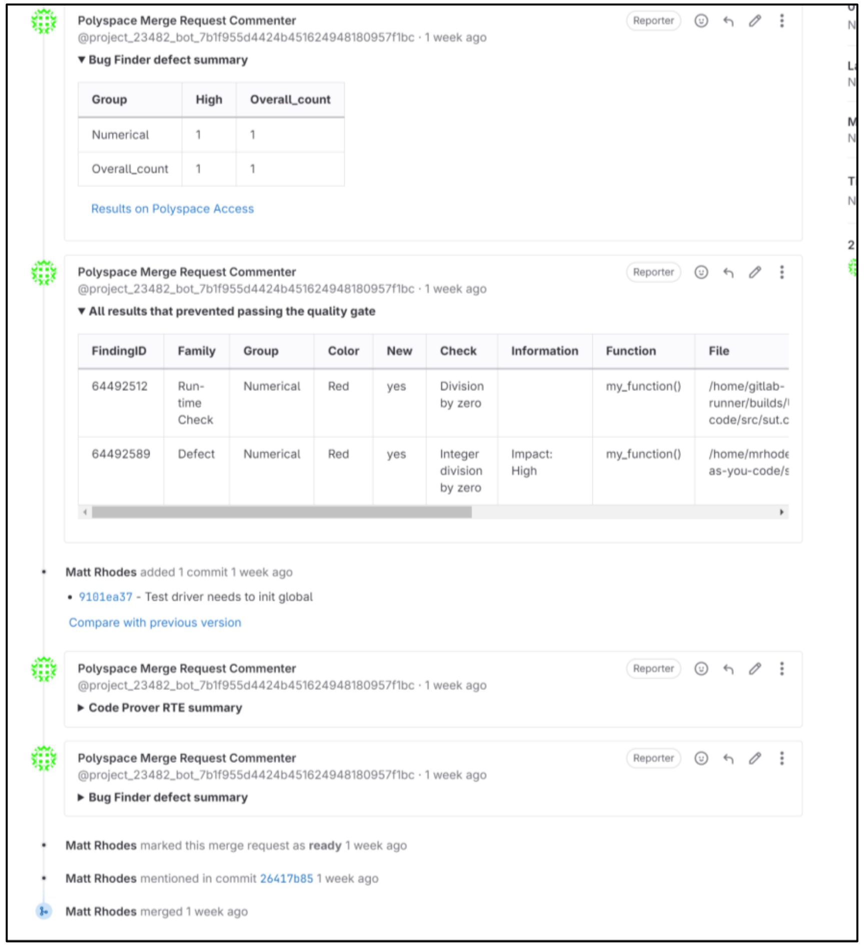
GitLab

Azure DevOps

Etc…

DEMO

Provide valuable feedback to your automation pipelines



Polyspace Merge Request Commenter
@project_23482_bot_7b1f955d4424b451624948180957f1bc · 1 week ago

▼ Bug Finder defect summary

Group	High	Overall_count
Numerical	1	1
Overall_count	1	1

Results on Polyspace Access

Polyspace Merge Request Commenter
@project_23482_bot_7b1f955d4424b451624948180957f1bc · 1 week ago

▼ All results that prevented passing the quality gate

FindingID	Family	Group	Color	New	Check	Information	Function	File
64492512	Run-time Check	Numerical	Red	yes	Division by zero		my_function()	/home/gitlab-runner/builds/ code/src/sut.c
64492589	Defect	Numerical	Red	yes	Integer division by zero	Impact: High	my_function()	/home/mrhode as-you-code/

Matt Rhodes added 1 commit 1 week ago

- 9101ea37 - Test driver needs to init global

Compare with previous version

Polyspace Merge Request Commenter
@project_23482_bot_7b1f955d4424b451624948180957f1bc · 1 week ago

▶ Code Prover RTE summary

Polyspace Merge Request Commenter
@project_23482_bot_7b1f955d4424b451624948180957f1bc · 1 week ago

▶ Bug Finder defect summary

Matt Rhodes marked this merge request as **ready** 1 week ago

Matt Rhodes mentioned in commit 26417b85 1 week ago

Matt Rhodes merged 1 week ago

Example of a pipeline posting custom merge request analysis results, and a quality gate result set showing blocking results. Both with links to the actual results on Polyspace Access.

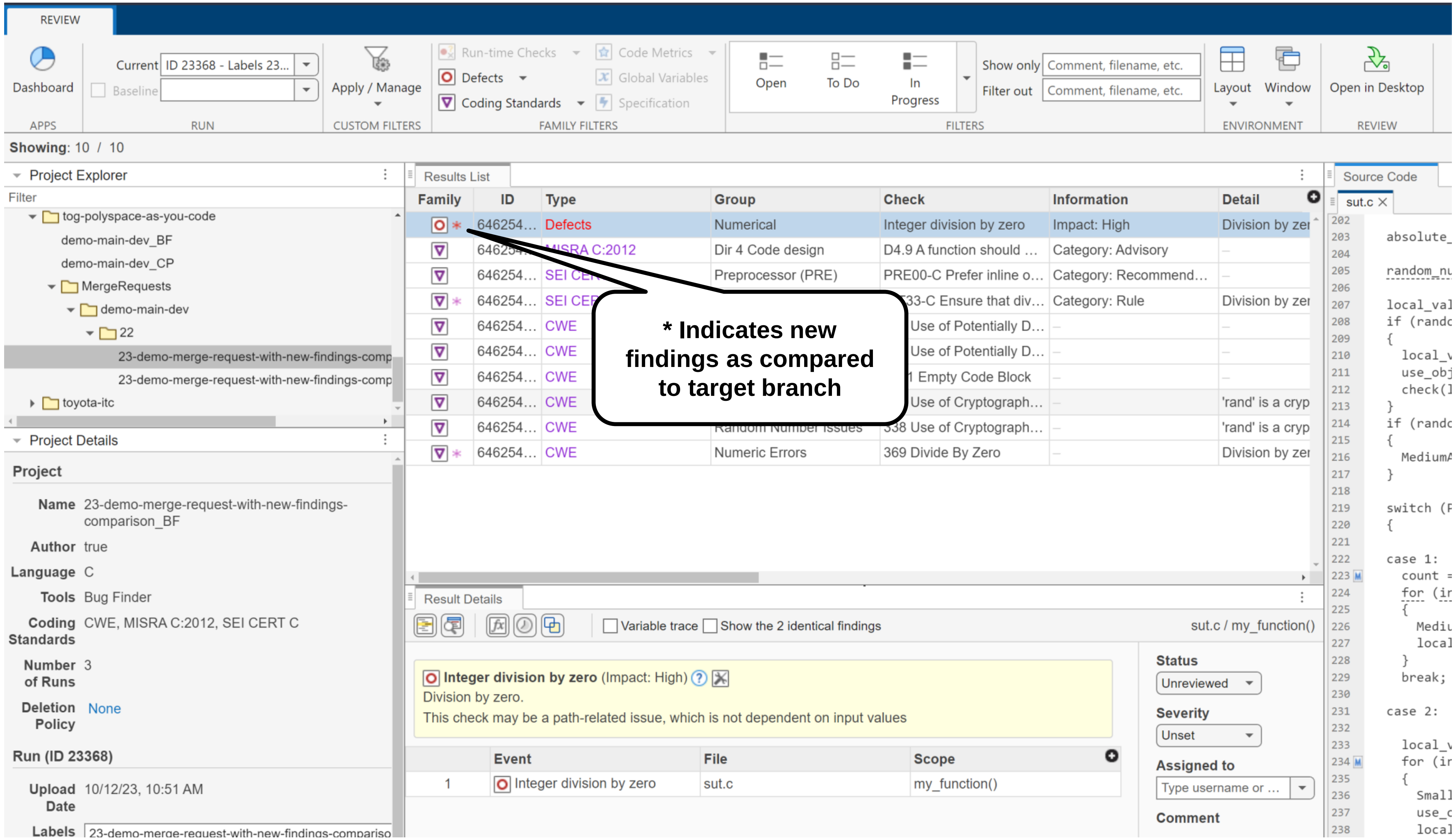
Results exports can be sent to CI pipeline comments via your CI system’s REST API

- Provide summaries
- Links to results view on Access
- Quality gates to “stop the line” or prevent a merge from adding unwanted bugs

Implementation

- Some reference implementations available by request
- Consulting support available

Results in Polyspace Access



Showing: 10 / 10

Project Explorer

Filter

- tog-polyspace-as-you-code
  - demo-main-dev_BF
  - demo-main-dev_CP
  - MergeRequests
    - demo-main-dev
      - 22
        - 23-demo-merge-request-with-new-findings-comp
        - 23-demo-merge-request-with-new-findings-comp
- toyota-itc

Project Details

Project

Name	23-demo-merge-request-with-new-findings-comparison_BF
Author	true
Language	C
Tools	Bug Finder
Coding Standards	CWE, MISRA C:2012, SEI CERT C
Number of Runs	3
Deletion Policy	None

Run (ID 23368)

Upload Date	10/12/23, 10:51 AM
Labels	23-demo-merge-request-with-new-findings-compariso

Results List

Family	ID	Type	Group	Check	Information	Detail
○ *	646254...	Defects	Numerical	Integer division by zero	Impact: High	Division by zer
▽	64625	MISRA C:2012	Dir 4 Code design	D4.9 A function should ...	Category: Advisory	–
▽	646254...	SEI CER	Preprocessor (PRE)	PRE00-C Prefer inline o...	Category: Recommend...	–
▽ *	646254...	SEI CER		33-C Ensure that div...	Category: Rule	Division by zer
▽	646254...	CWE		Use of Potentially D...	–	–
▽	646254...	CWE		Use of Potentially D...	–	–
▽	646254...	CWE		1 Empty Code Block	–	–
▽	646254...	CWE		Use of Cryptograph...	–	'rand' is a cryp
▽	646254...	CWE	Random Number Issues	338 Use of Cryptograph...	–	'rand' is a cryp
▽ *	646254...	CWE	Numeric Errors	369 Divide By Zero	–	Division by zer

*** Indicates new findings as compared to target branch**

Result Details

Variable trace   Show the 2 identical findings   sut.c / my_function()

Integer division by zero (Impact: High)
Division by zero.
This check may be a path-related issue, which is not dependent on input values

	Event	File	Scope
1	Integer division by zero	sut.c	my_function()

Status Unreviewed

Severity Unset

Assigned to Type username or ...

Comment

Source Code

sut.c

```
    absolute_

    random_nu

    local_val
    if (rando
    {
      local_v
      use_obj
      check(l
    }
    if (rando
    {
      MediumA
    }

    switch (P
    {

    case 1:
      count =
      for (ir
      {
        Mediu
        local
      }
      break;

    case 2:

      local_v
      for (ir
      {
        Small
        use_c
        local
```

]

Wrap-up

Which verification effort is easier?

Only testing	Using Polyspace
• Significant uncertainty • More churn from later development stages • Extra work for certification processes	• Automated Proof • Avoid needless churn with Pre-integration analysis • Credits for certification processes

Implementation Plan and Support Options

Getting Started (free)

- License & Install Support
- Online documentation, incl:
  - Getting started, what's new, examples, context-sensitive help, …
- AE-guided "quick start"
- Technical Support

Training (add'l cost)

- End Users (1 Day)
  - Create effective reviewers
- Power Users (2-3 Day)
  - Create experts able to set up analysis and support others

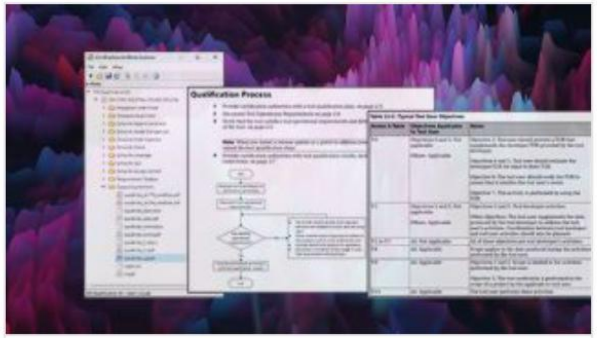
Consulting (add'l cost)

- Advanced setup
- Integrations
- Scripting and automation
- Report customization
- Etc.

Ongoing Support (free)

- Technical Support
- Awareness building: seminars, lunch-n-learns, workshops, etc.
- Check-ins, "What's new" briefings
- AE "Office Hours"
- Etc.

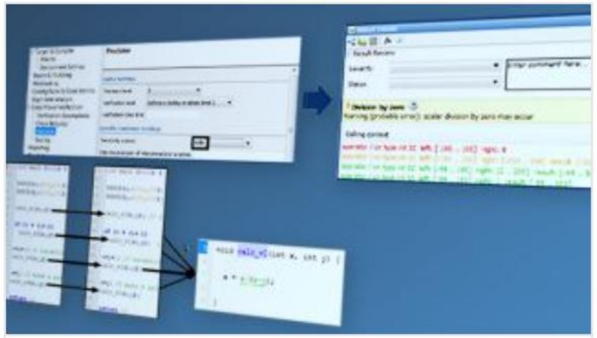
Training Courses



Model-Based Design for DO-178C/DO-331 Compliance

Build on prior knowledge of Simulink modeling principles and verification workflows in Simulink and Polyspace to generate production code intended for DO-178C certification.

ADVANCED



Polyspace for C/C++ Code Verification

Prove code correctness, review and understand verification results, handle missing functions and data, measure software quality metrics, and apply MISRA C rules.

INTERMEDIATE

Reviewing Polyspace Results

Interpret Polyspace Bug Finder and Polyspace Code Prover results in Polyspace Access to remove algorithmic defects, improve software quality metrics, and improve product integrity.

INTERMEDIATE

Q & A