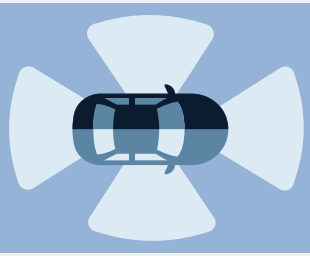
MathWorks
**AUTOMOTIVE CONFERENCE 2022**
North America

# Faster Software Delivery with Polyspace in Your Software Factory

*Patrick Munier, Software Development Director, MathWorks*

MathWorks®

# Software Lines of Code and Delivery Frequency are increasing

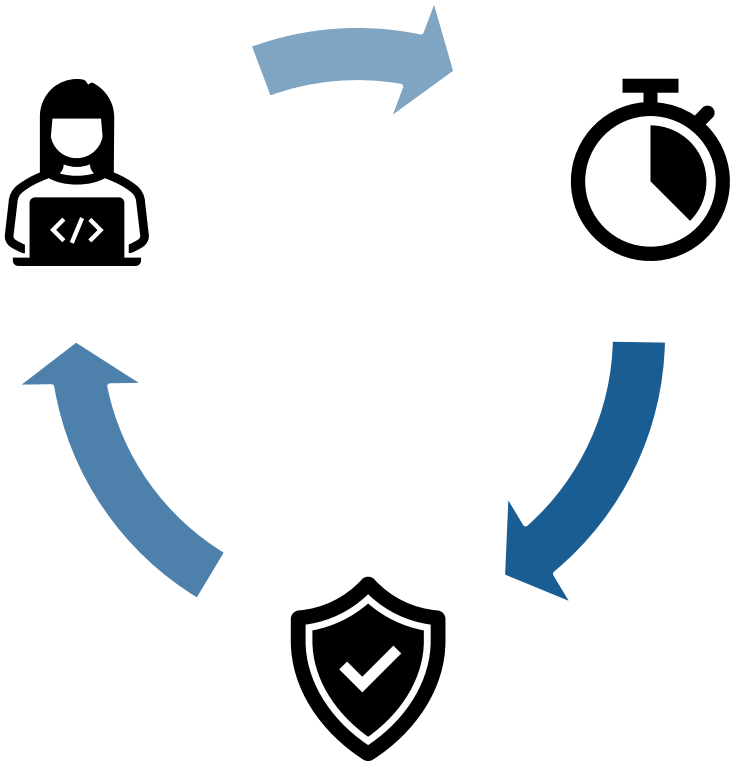
| 1980 | 2000 | 2020 | Future |
|---|---|---|---|
|  |  |  |  |
| 10s of KLOC | 100s of K | 100s of M | >500M |
| ~5 years / release | 3 – 5 years | 0.4 – 1 year | On-demand |

# Software Development in Embedded systems is facing major challenges

Size and complexity

C++, AUTOSAR

Staffing SW developers

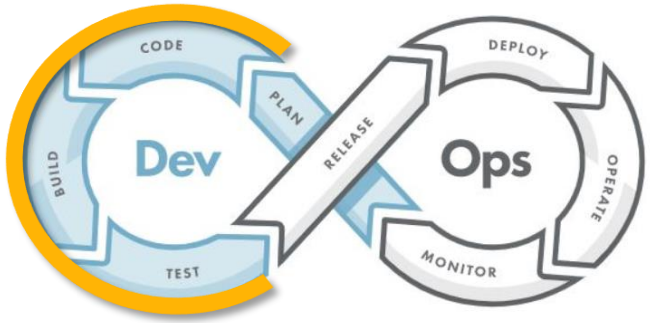


Faster delivery

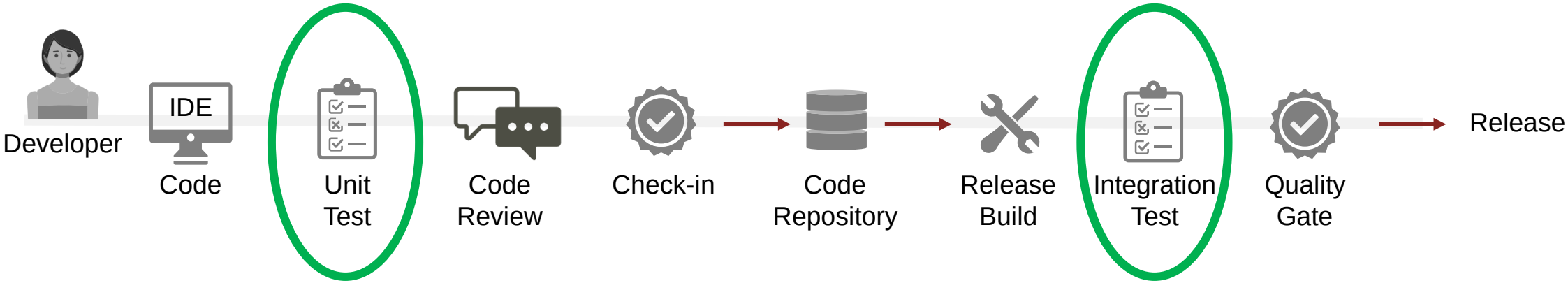
Safety and Cybersecurity

ISO 26262, ISO/SAE 21434

# Software Factory help solve these challenges

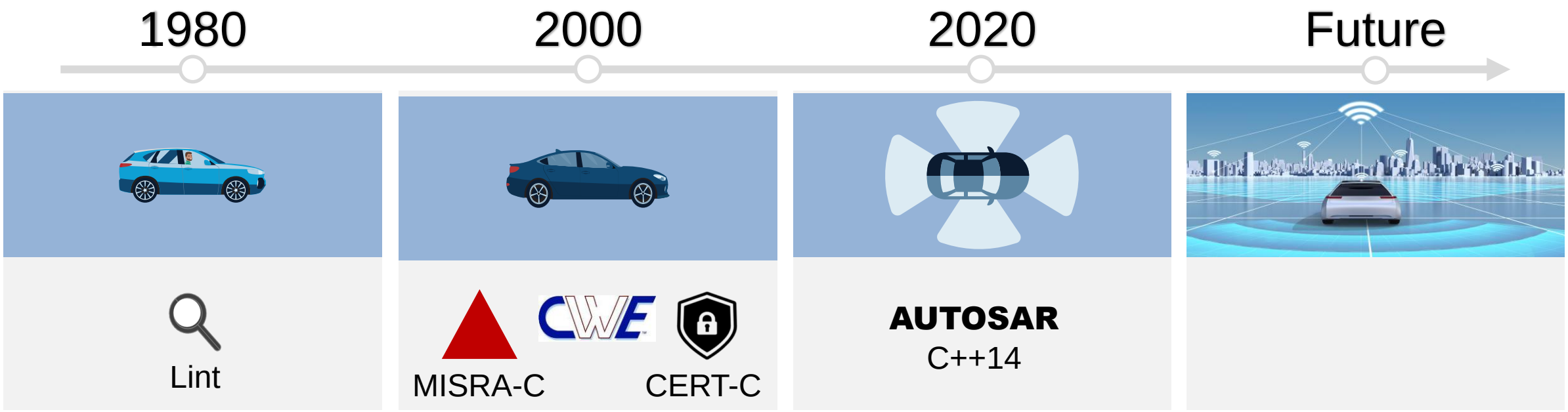



**DevOps** is a set of practices that combines software development (Dev) and IT operations (Ops)




1. Repeatability
2. Faster delivery
3. Higher quality

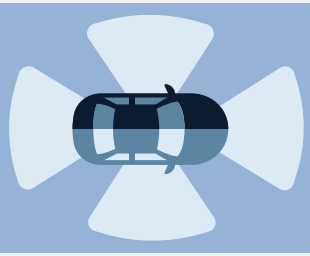
# Static Analysis analyzes code without the need of tests to execute



# Static Analysis is the most effective way to check compliance with Coding Rules

## 5.3 Compliance

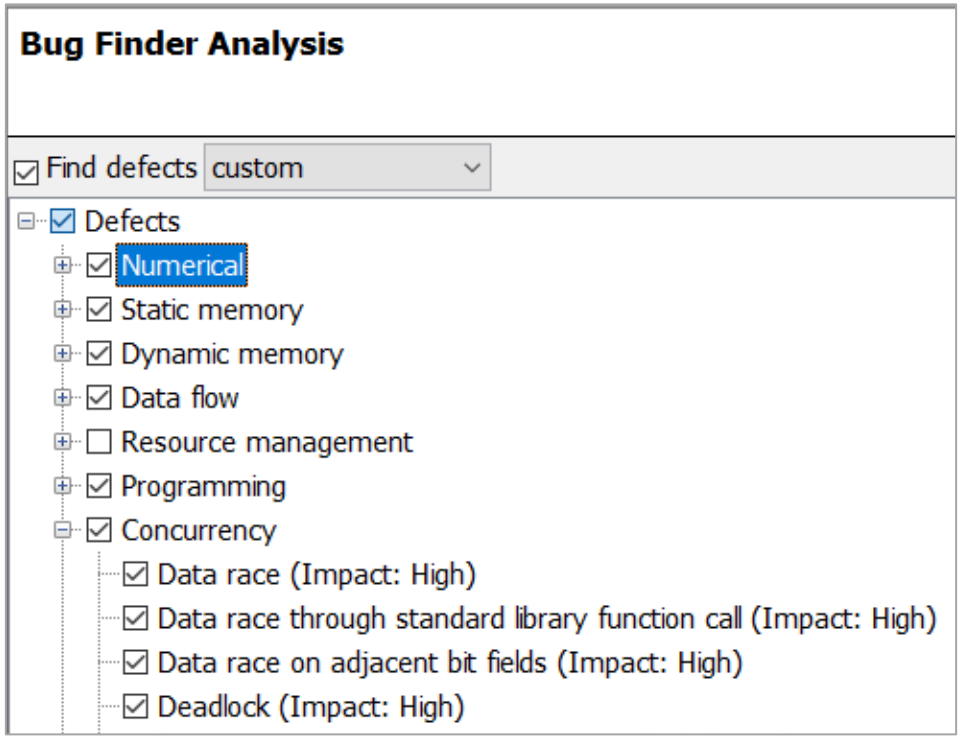
In order to ensure that code complies with all of the MISRA C guidelines, a compliance matrix should be produced. This matrix lists each guideline and indicates how it is to be checked. For most guidelines, the easiest, most reliable and most cost-effective means of checking will be to use a static analysis tool or tools, the compiler, or a combination of these. Where a guideline cannot be completely checked by a tool, then a manual review will be required.

| 1980 | 2000 | 2020 | Future |
|---|---|---|---|
|  |  |  |  |
| Lint | MISRA-C, CERT-C | **AUTOSAR** C++14 | |

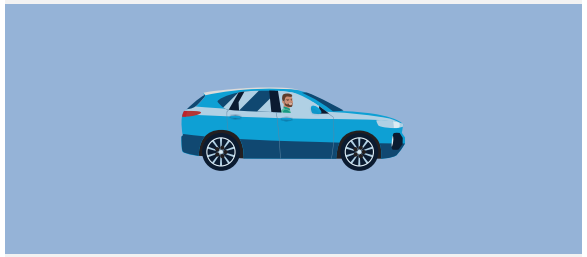
# Static Analysis finds various kinds of defects

Ada C C++

**Polyspace**

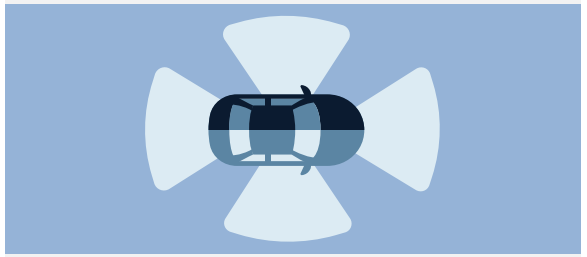



1980 2000



Lint

MISRA-C CERT-C



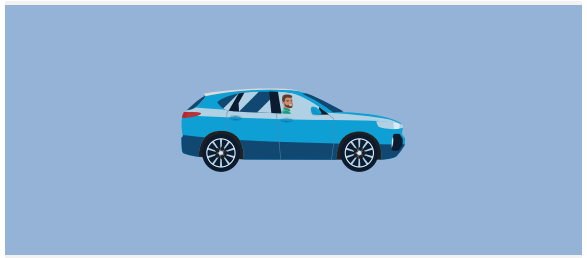
**AUTOSAR**
C++14

# Static Analysis proves the absence of critical runtime errors

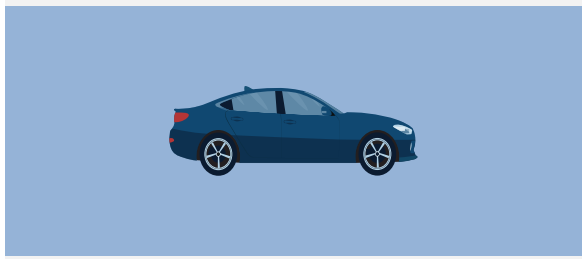
Ada C C++

**Polyspace**

1980 2000



Lint



MISRA-C CERT-C

**Green: reliable**
safe pointer access

**Red: faulty**
out of bounds error

**Gray: dead**
unreachable code

**Orange: unproven**
may be unsafe for some conditions

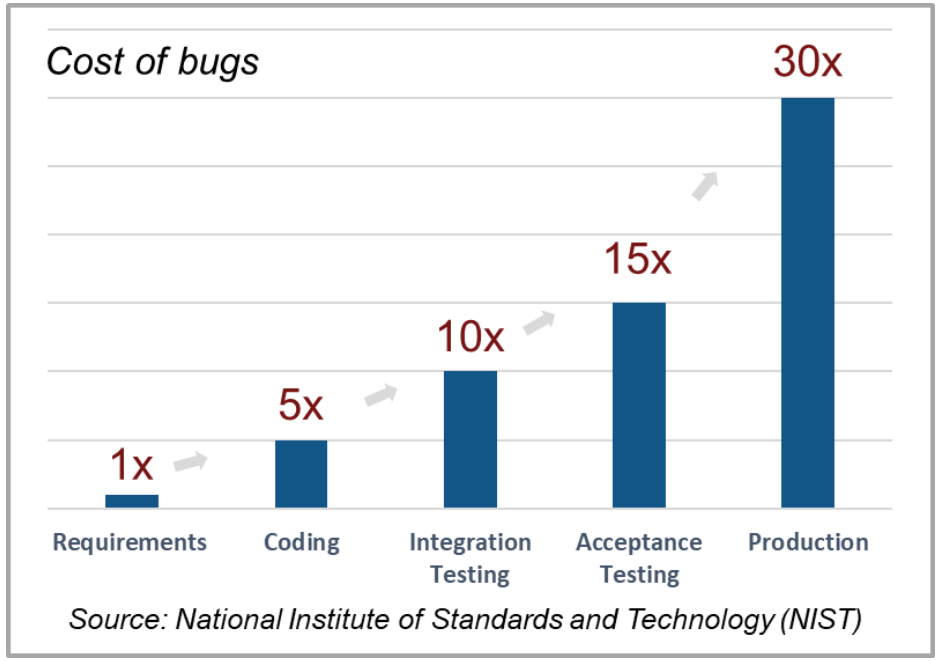
**Purple: violation**
MISRA-C/C++ or JSF++ code rules

***Range data***
*tool tip*

```
static void pointer_arithmetic (void) {
    int array[100];
    int *p = array;
    int i;

    for (i = 0; i < 100; i++) {
      *p = 0;
      p++;
    }

  if (get_bus_status() > 0) {
    if (get_oil_pressure() > 0) {
      *p = 5;
    } else {
      i++;
    }
  }

  i = get_bus_status();

  if (i >= 0) {
    *(p - i) = 10;
  }
}
```

variable 'I' (int32): [0 .. 99]
assignment of 'I' (int32): [1 .. 100]

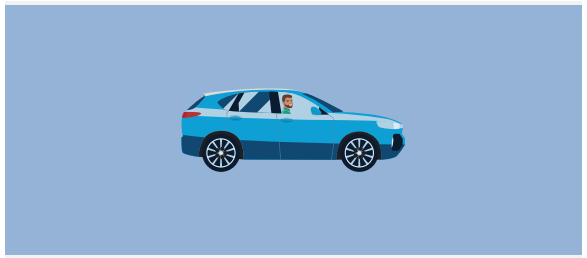
# Static Analysis should be used early in the development process




Ada C C++

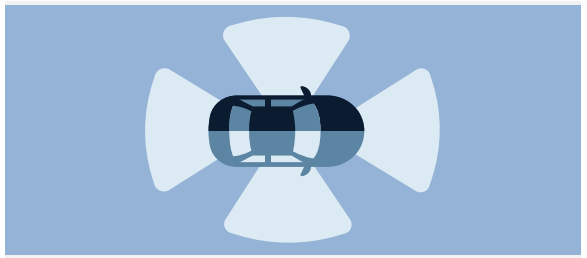
**Polyspace**

1980 2000 2020 Future



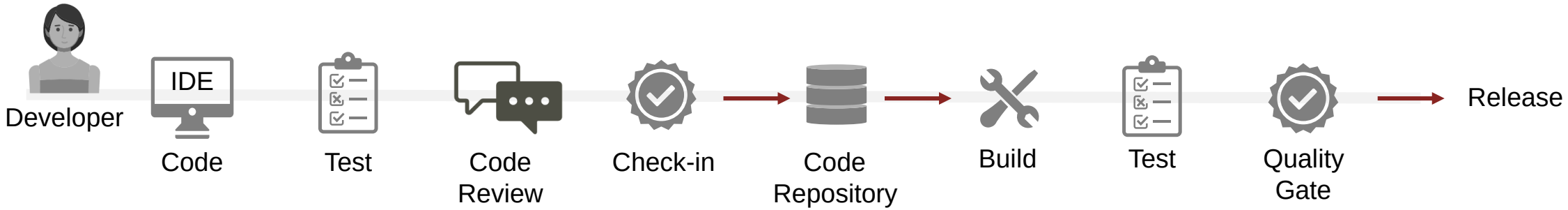
Lint

MISRA-C CERT-C



**AUTOSAR**
C++14

# Static Analysis in Software Factories provides many benefits




## 1. Repeatability

Find errors and verify coding standards consistently

## 2. Faster delivery

Find defects early

Reduce testing

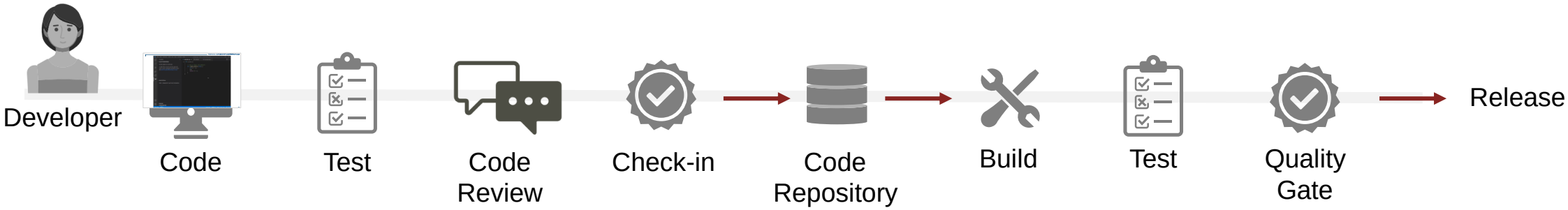
Upskill Developers

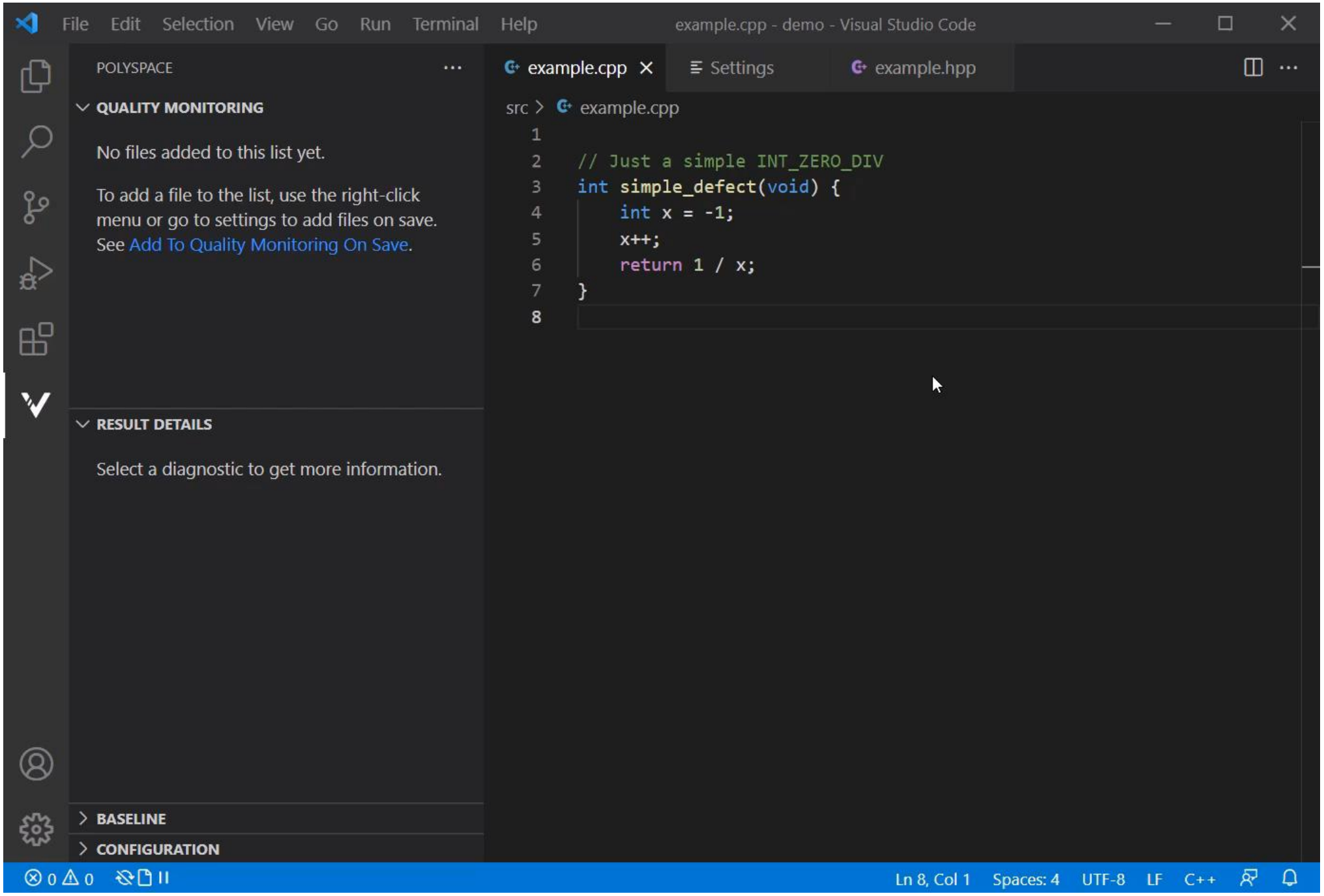
## 3. Higher Quality

Quality gates

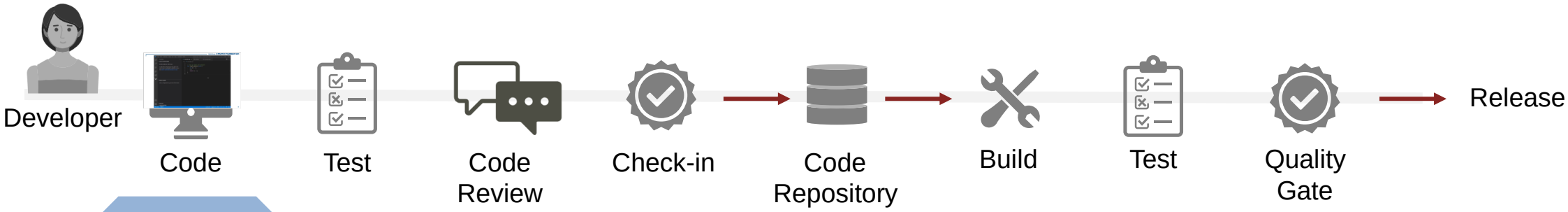
Including the absence of critical runtime errors!

# Static Analysis can be integrated as early as in the IDE



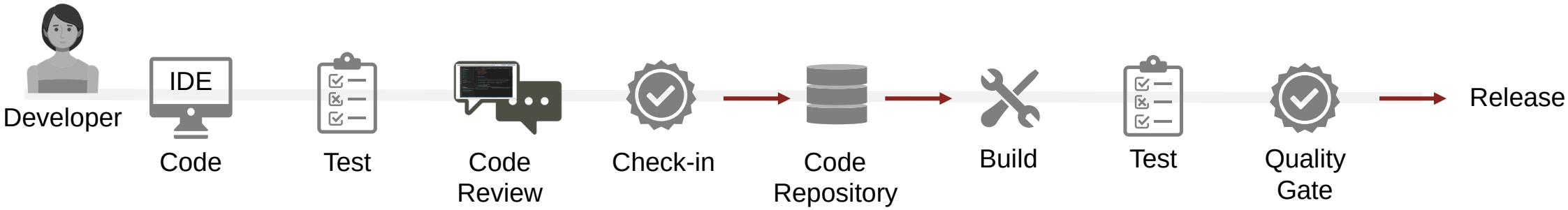

File Edit Selection View Go Run Terminal Help
example.cpp - demo - Visual Studio Code
POLYSPACE
QUALITY MONITORING
No files added to this list yet.
To add a file to the list, use the right-click menu or go to settings to add files on save.
See Add To Quality Monitoring On Save.
RESULT DETAILS
Select a diagnostic to get more information.
BASELINE
CONFIGURATION
example.cpp
Settings
example.hpp
src > example.cpp
1
2 // Just a simple INT_ZERO_DIV
3 int simple_defect(void) {
4 int x = -1;
5 x++;
6 return 1 / x;
7 }
8
Ln 8, Col 1 Spaces: 4 UTF-8 LF C++

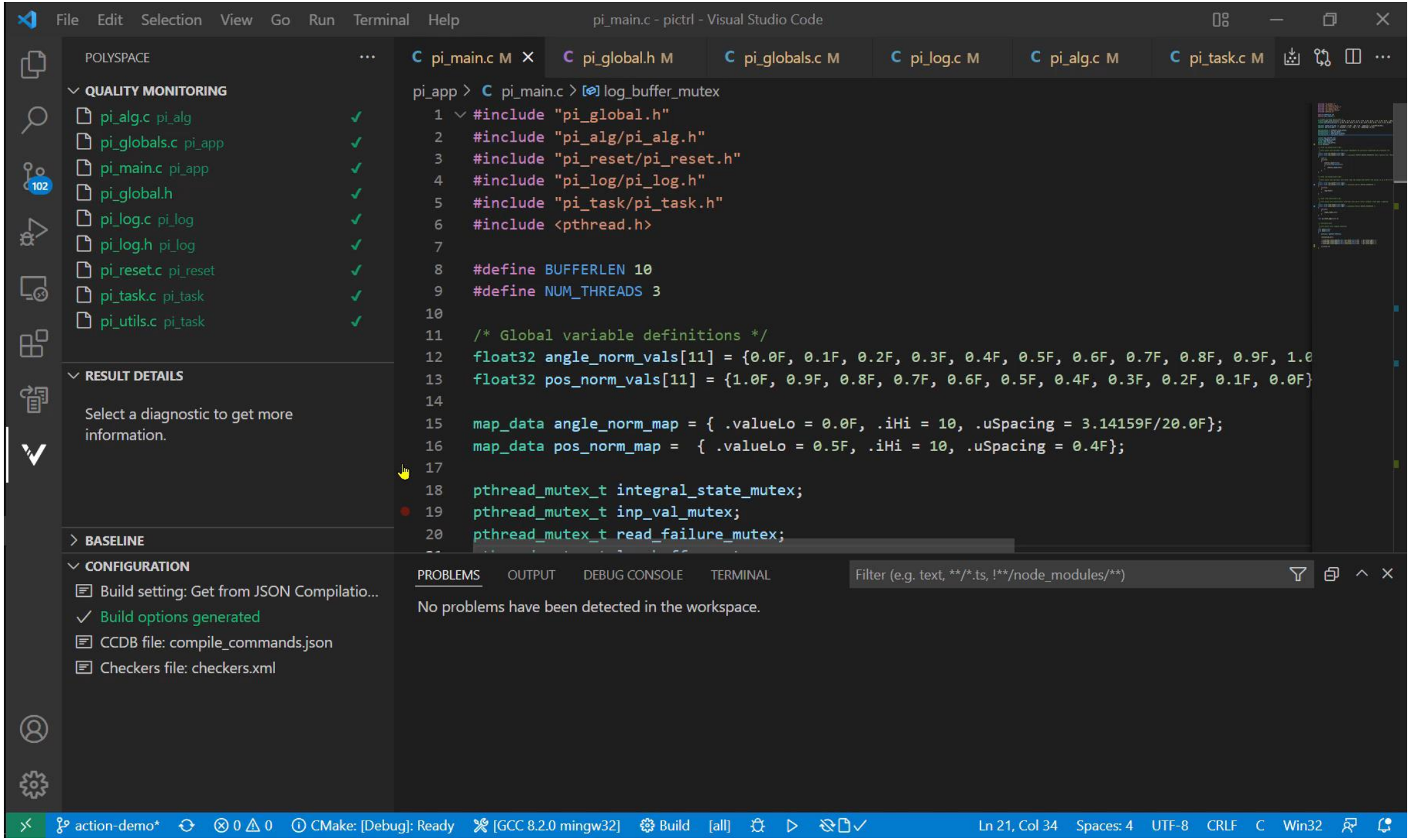
# Polyspace as You Code finds defects early and helps train developers



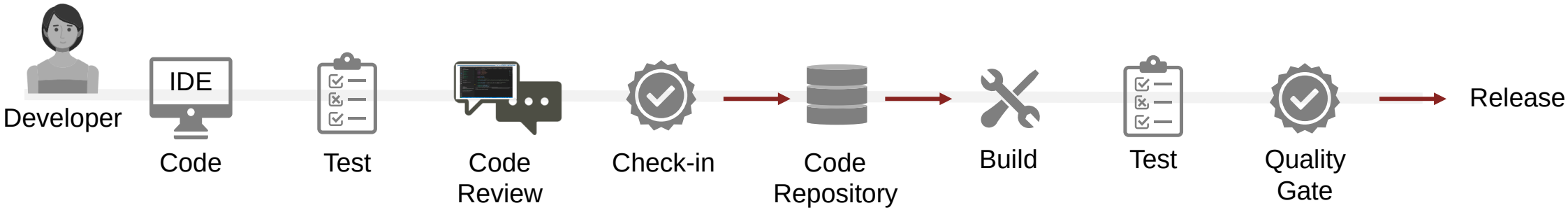

- Find and fix defects as soon as code is written
- Instant feedback, enables continuous training

# Static Analysis can be a verification bot in the Code Review



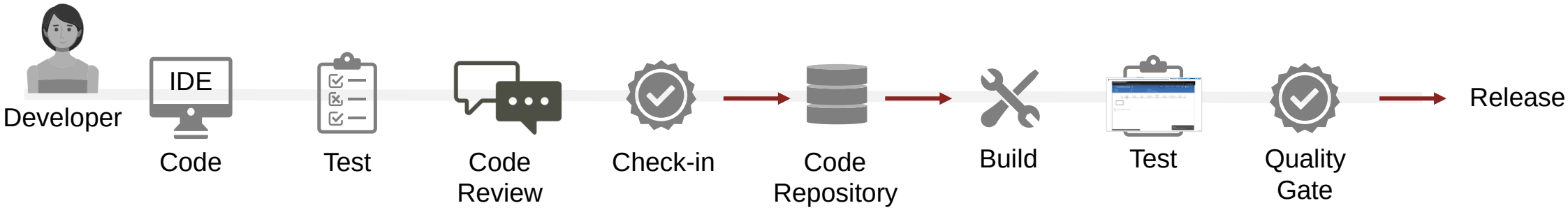

File Edit Selection View Go Run Terminal Help
pi_main.c - pictrl - Visual Studio Code
POLYSPACE
QUALITY MONITORING
pi_alg.c pi_alg
pi_globals.c pi_app
pi_main.c pi_app
pi_global.h
pi_log.c pi_log
pi_log.h pi_log
pi_reset.c pi_reset
pi_task.c pi_task
pi_utils.c pi_task
RESULT DETAILS
Select a diagnostic to get more information.
BASELINE
CONFIGURATION
Build setting: Get from JSON Compilatio...
Build options generated
CCDB file: compile_commands.json
Checkers file: checkers.xml
pi_main.c M
pi_global.h M
pi_globals.c M
pi_log.c M
pi_alg.c M
pi_task.c M
pi_app > pi_main.c > log_buffer_mutex
#include "pi_global.h"
#include "pi_alg/pi_alg.h"
#include "pi_reset/pi_reset.h"
#include "pi_log/pi_log.h"
#include "pi_task/pi_task.h"
#include <pthread.h>
#define BUFFERLEN 10
#define NUM_THREADS 3
/* Global variable definitions */
float32 angle_norm_vals[11] = {0.0F, 0.1F, 0.2F, 0.3F, 0.4F, 0.5F, 0.6F, 0.7F, 0.8F, 0.9F, 1.0
float32 pos_norm_vals[11] = {1.0F, 0.9F, 0.8F, 0.7F, 0.6F, 0.5F, 0.4F, 0.3F, 0.2F, 0.1F, 0.0F}
map_data angle_norm_map = { .valueLo = 0.0F, .iHi = 10, .uSpacing = 3.14159F/20.0F};
map_data pos_norm_map = { .valueLo = 0.5F, .iHi = 10, .uSpacing = 0.4F};
pthread_mutex_t integral_state_mutex;
pthread_mutex_t inp_val_mutex;
pthread_mutex_t read_failure_mutex;
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Filter (e.g. text, **/*.ts, !**/node_modules/**)
No problems have been detected in the workspace.
action-demo* 0 0 CMake: [Debug]: Ready [GCC 8.2.0 mingw32] Build [all]
Ln 21, Col 34 Spaces: 4 UTF-8 CRLF C Win32

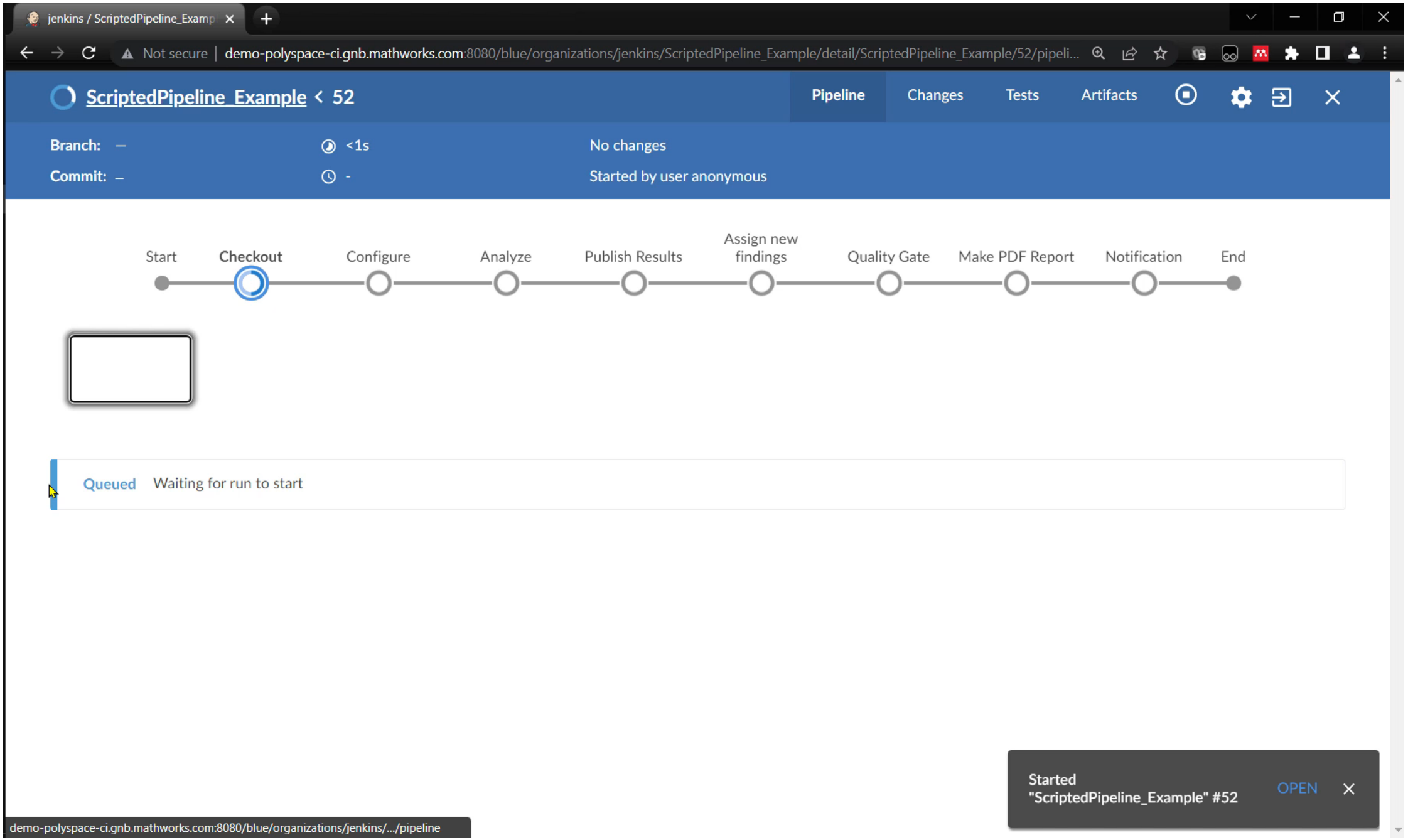
# Polyspace automates (a boring part of) the Code Reviews



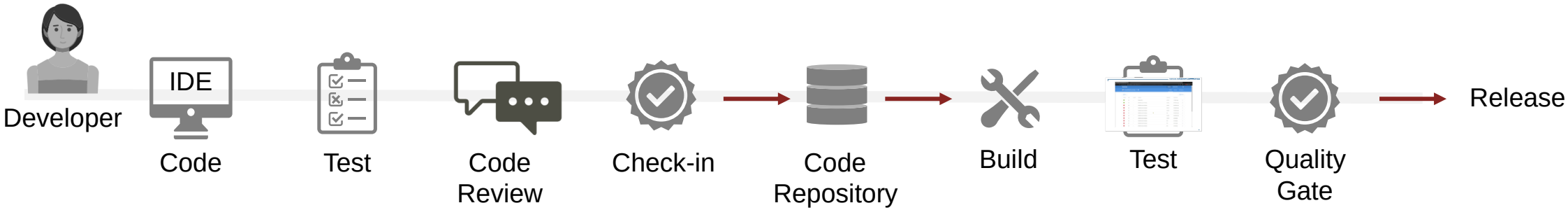

- Automate tedious code review tasks
- Help developers focus on algorithms

# Static Analysis should be integrated in a CI pipeline




jenkins / ScriptedPipeline_Examp
Not secure | demo-polyspace-ci.gnb.mathworks.com:8080/blue/organizations/jenkins/ScriptedPipeline_Example/detail/ScriptedPipeline_Example/52/pipeli...
ScriptedPipeline_Example < 52
Pipeline
Changes
Tests
Artifacts
Branch: –
Commit: –
<1s
-
No changes
Started by user anonymous
Start
Checkout
Configure
Analyze
Publish Results
Assign new findings
Quality Gate
Make PDF Report
Notification
End
Queued
Waiting for run to start
Started "ScriptedPipeline_Example" #52
OPEN
demo-polyspace-ci.gnb.mathworks.com:8080/blue/organizations/jenkins/.../pipeline

# Polyspace assesses Software Quality vs Quality Gates




- Define consistent Quality Gates
- Evaluate final quality of code
- Automatically generate reports

# Conclusion – Future direction…


Obviously developed with a Software Factory

… with heavy use of Static Analysis from IDE to CI!

To help you apply Static Analysis…

We'll continue to evolve Polyspace!

# Thank you