

# MATLAB EXPO 2019

Making Software Safe and Secure  
with Team Collaboration

Vaishnavi H R  
Application Engineer, Mathworks



# Agenda

1. Making Software Safe and Secure
2. Polyspace Static Analysis
3. Team Collaboration with Polyspace

# 1. Making Software Safe and Secure

**“Program testing can be used to show the presence of bugs,  
but never to show their absence”**

**Edsger Dijkstra**, Computer Science Pioneer

**“Given that we cannot really show there are no more errors  
in the program, when do we stop testing?”**

**Brent Hailpern**, Head of Computer Science

# Using Static Analysis to Make Software Safe and Secure

- Find bugs without code execution
  - Code analyzed without running tests
  - Identify bugs and coding rule violations for MISRA, AUTOSAR, CERT
- Prove absence of critical run-time errors
  - Identify code that will never experience errors regardless of run-time conditions
- Complements dynamic testing
  - Used together, you can find more bugs for higher quality code

```

main.cpp x
20
21 static bool table_loop(void)
22 {
23     int j = 4;
24
25     // Table of basic element
26     Base* array[] = { new SAnalogic, new Sensor, new Sensor, new SAnalogic };
27
28     for (int i = 4; i >= 0; i--, j--) {
29         array[i-1]->Draw();
30
31         // Error for the 2 last elements: this cast is similar to static_cast
32         // the TypeInfo function only define in SAnalogic
33         if (i % 2)
34             ((SAnalogic*)(array[i-1]))->TypeInfo();
35         else
36             (dynamic_cast<SAnalogic*>(array[i-1]))->TypeInfo();
37     }

```

	Event	File	Scope
1	Iterating on loop	main.cpp	table_loop()
2	This-pointer of TypeInfo is null	main.cpp	table_loop()
3	● Non-terminating loop	main.cpp	table_loop()

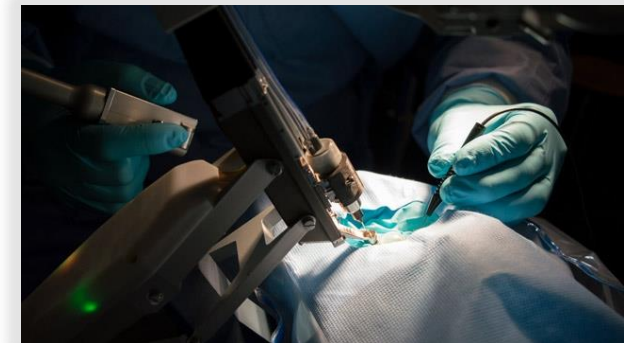
## ● Non-terminating loop ?

The loop is infinite or contains a run-time error.

Loop fails due to a run-time error (maximum number of iterations: 3).

# When Software Safety and Security Matter

- Industries where safety and security matter
  - Automotive, Aerospace, Medical Device, Industrial Machinery
- Governed by functional safety and other standards
  - ISO 26262, DO-178, IEC 62304, IEC 61508
  - MISRA, CERT, AUTOSAR
- Static analysis provides certification credits
  - For standards such as ISO 26262 and DO-178



## 2. Polyspace Static Analysis

*For software written in C, C++, and Ada*

# Proving Absence of Critical Run-Time Errors

```
float x, y;  
  
...  
  
x = x / (x - y);
```

- How many run-time errors are possible?
  1. Divide by zero
  2. Overflow
  3. Uninitialized variables
- How to test all floating point variable combinations?
- How do you prove that this code will not fail?



# Proving Absence of Critical Run-Time Errors

Proven by Polyspace that  
run-time error will not occur

## ✓ Division by zero ?

Float division by zero does not occur  
operator / on type float 32

left: 10.0

right: [-31.1328 .. -11.1327]

result: [-0.89826 .. -0.3212]

```

1  float where_are_errors_float(float input)
2  {
3      float x, y, k, l, limit = 1000.0f;
4
5      if (input < -limit || input > limit) return (-9999.0f);
6
7      k = input / 100.0f;
8      x = 2.0f;
9      y = k + 5.0f;
10
11     while (x < 10.0f)
12     {
13         x++;
14         y = y + 3.141592f;
15     }
16
17     if ((3.0*k + 100.0f) >= 71.0f)
18     {
19         y++;
20         x = x / (x - y);
21     }
22
23     return x;
24 }

```

# Proving Absence of Critical Run-Time Errors with Polyspace

The screenshot displays the Polyspace web interface. The browser address bar shows the URL `localhost:9443/metrics/index.html?a=review&p=3&r=2`, which is circled in red. The interface includes a top navigation bar with tabs for Polyspace, Polyspace Access Cluster Operate, and System Dashboard - Jira. Below this is a navigation menu with options like Dashboard, Run-time Checks, Defects, Coding Standards, Code Metrics, Global Variables, To Do, In Progress, Done, Filter out, Comment, filename, etc., Layout, and Open in Desktop. The main content area shows a table of defects with columns for ID, Type, Group, Check, and Information. A specific defect (ID 117) is highlighted, showing a 'Green Check' for 'Division by zero'. A callout box labeled 'Defect Details' points to this row. Another callout box labeled 'Filter Results' points to the 'Filter out' button. A third callout box labeled 'Source Code View' points to the 'Source Code' tab, which displays the C code for the function `where_are_the_errors_float`. A fourth callout box labeled 'Defect List' points to the 'Results List' tab, which shows the table of defects. The bottom of the interface shows the file explorer and project details for the file `Where_Are_Errors_Float2`.

Polyspace Access Cluster Operate System Dashboard - Jira

localhost:9443/metrics/index.html?a=review&p=3&r=2

Dashboard Run-time Checks Defects Coding Standards Code Metrics Global Variables To Do In Progress Done Filter out Comment, filename, etc. Layout Open in Desktop

APPS FAMILY FILTERS FILTERS ENVIRONMENT REVIEW

Showing: 62 / 62

Results List

ID	Type	Group	Check	Information	Detail
72	Code Metrics	File Metrics	Comment Density	Value: 10	
70	Code Metrics	Function Metrics	Cyclomatic Complexity	Value: 4	
96	Green Check	Numerical	Division by zero	-	
117	Green Check	Numerical	Division by zero	-	

Result Details

Status: Unreviewed Severity: Unset Assigned to: Type username or... Track issue Create Ticket

Defect Details

Filter Results

Source Code View

Defect List

Source Code

```
where_are_the_errors.c / where_are_errors_float()
1: float where_are_errors_float(float input)
2: {
3:     float x = input;
4:     x = 2.0f;
5:     y = k + 5.0f;
6:     while (x < 10.0f)
7:     {
8:         y++;
9:         x = x / (x - y);
10:     }
11:     return x;
12: }
```

Where\_Are\_Errors\_Float2

# Polyspace Tools

## Bug Finder

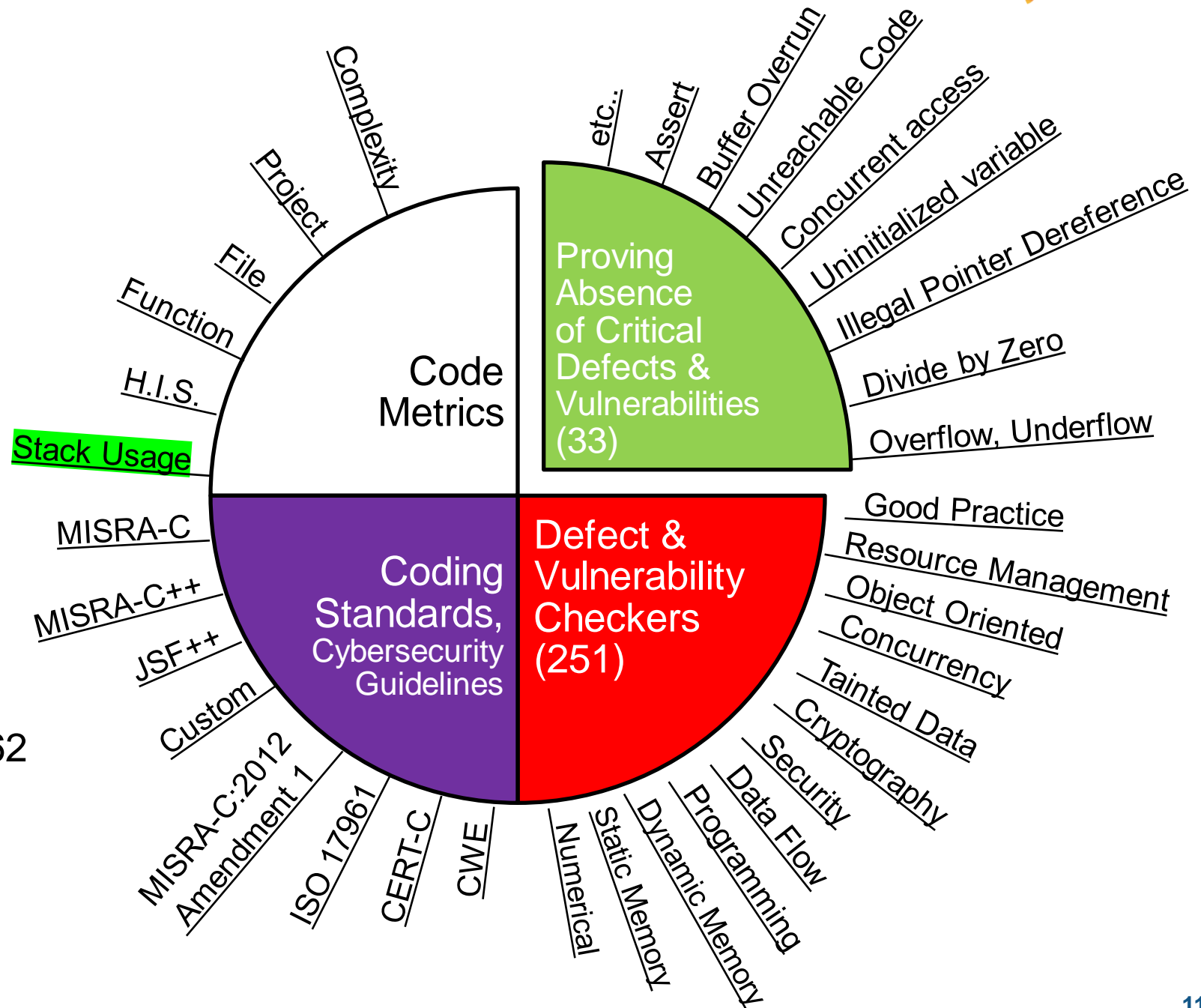


- Produce code metrics
- Check coding standards
- Find defects and vulnerabilities

## Code Prover



- Proves code Safe and Secure
- 33 most critical run-time checks
- Supports DO-178 and ISO 26262



# Polyspace Customer References



Electronic Steering Lock

KOSTAL Asia R&D Center Receives ISO 26262 ASIL D Certification for Automotive Software



Alenia Aermacchi Develops Autopilot Software for DO-178B Level A Certification

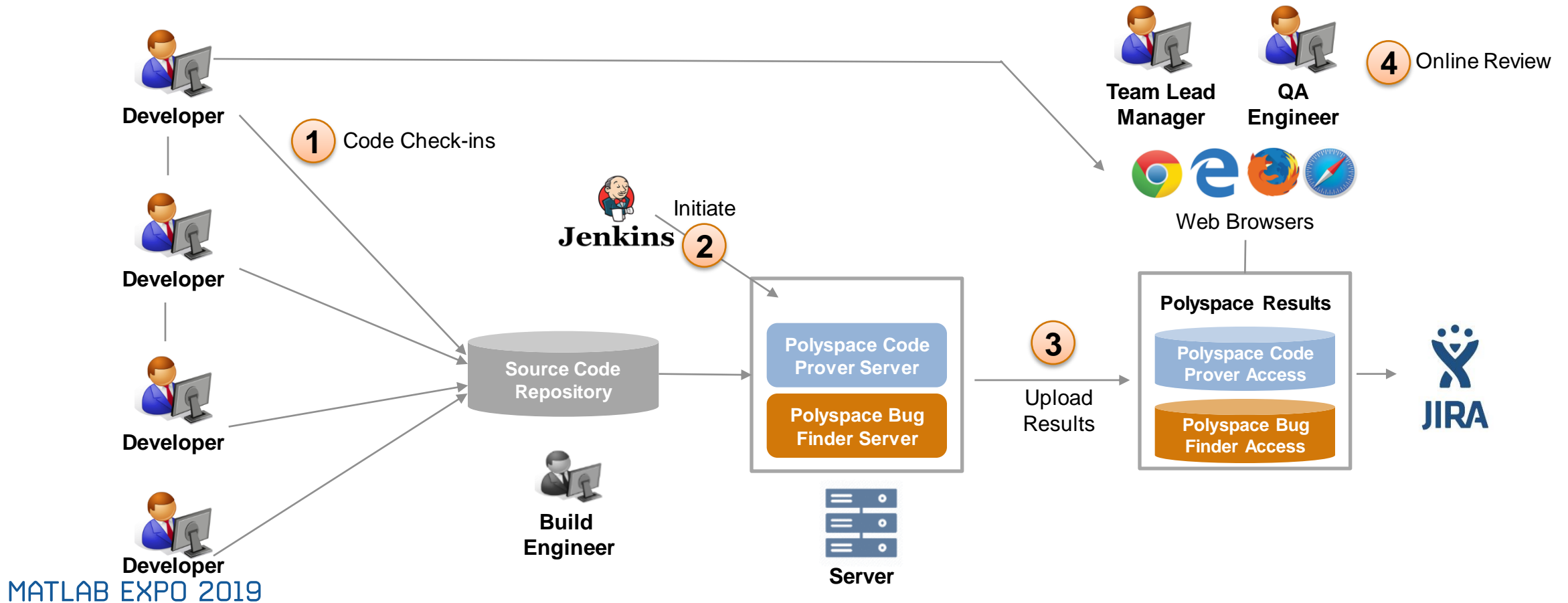


Miracor Eliminates Run-Time Errors and Reduces Testing Time for Class III Medical Device Software

## 3. Team Collaboration with Polyspace

# Workflow with New Polyspace Products in R2019a

1. Developers check-in code into repository, Build Engineer has configured Jenkins to run Polyspace analysis
2. Jenkins initiates Polyspace analysis run on the server (periodically or at program milestones)
3. Once Polyspace analysis run concludes, results are uploaded to Polyspace Access
4. Team Lead/Manager, QA, Developers use web browser to review results, open Jira defects, monitor quality metrics



# Team Collaboration Story



Bob is the Build Engineer  
He has configured Polyspace in a Jenkins CI workflow



Quinn is a Quality Engineer  
She is responsible for triaging software defects



Dara is a software developer  
She is responsible for writing code and fixing defects



Eric is a Simulink and Embedded Coder user  
He is responsible for generating code from models



Martin is a project manager  
He is responsible for software quality of the project



Bob is the Build Engineer  
He has configured Polyspace in a Jenkins CI workflow

**Jenkins** 4 search

Jenkins > BF\_POLYSPACE\_LANG\_MODULES > #35

- Back to Project
- Status
- Changes
- Console Output
- View as plain text
- Edit Build Information
- Delete Build
- Previous Build
- Next Build

### Console Output

```
Starting at: Tue Feb 12 02:41:11
Host: Linux cpu-02-ah 4.9.0-8-a
User: jenkins
*****
*** Beginning Bug-finder - Modu
***
*****
**** Bug-finder - Module Analys
* Created 2 modules
**** Bug-finder - Module Analys
**** Bug-finder - Module Analys
**** Bug-finder - Module Analys
**** Bug-finder - Module Analys
**** Bug-finder - Module Analys
Maximum Memory Usage: 2913 MB

Generating GUI files

Defects statistics:
- Total number of defects: 46
- ASSERT: 2
- MEM_LEAK: 2
- NON_INIT_PTR: 1
- UNPROTECTED_MEMORY_ALLOCATI
- USELESS_WRITE: 1
```

Polyspace Access Cluster Operator

localhost:8080/services

### Polyspace Access Cluster Operator

- Services
- Nodes
- Settings

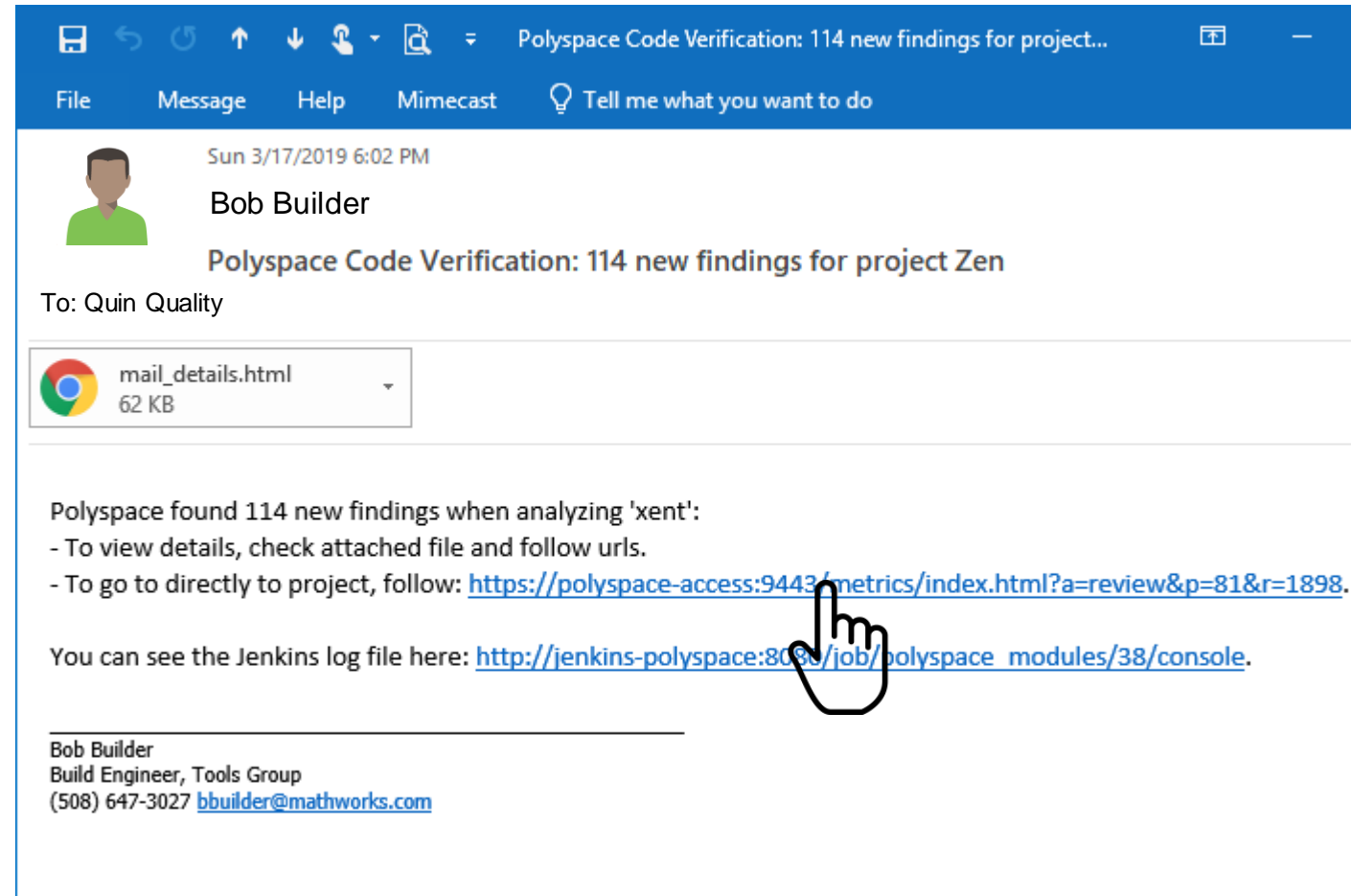
Services				
	PROVISION	START ALL	STOP ALL	DELETE ALL
User Manager	●	Running	Stop	
Database	●	Running	Stop	
ETL	●	Running	Stop	
Web Server	●	Running	Stop	
Gateway	●	Running	Stop	





Quinn is a Quality Engineer  
She is responsible for triaging software defects

- She received an email notification from last night's Jenkins initiated Polyspace analysis
- The email indicates several findings were found in her project
- She click on the link in the email to view the findings in Polyspace Access






Quinn is a Quality Engineer  
She is responsible for triaging software defects

Sign In

localhost:9443/authn/signin?&title=/static/images/polyspace\_title\_logo.svg&continue=%2F



**Sign in to your account**

[Forgot password?](#)

**Sign in**



Dara is a software developer  
She is responsible for writing code and fixing defects

- Dara has been assigned 2 defect tickets in Jira
- She opens the first JIRA ticket and clicks the Polyspace Access link

The screenshot shows a web browser window with the URL `https://jira-test-aws.mathworks.com/browse/UXVIZ-620`. The page header includes the MathWorks logo and navigation links for Jira, Dashboards, Projects, Issues, Boards, Structure, and MathWorks Applications. A 'Create' button is visible in the top right.

The main content area displays the 'Project Zen' sidebar on the left and the ticket details on the right. The ticket title is 'Illegally dereferenced pointer' under the 'Visual Design' project. The ticket status is 'TO DO' with a '(View Workflow)' link.

The 'Details' section shows the following information:

Field	Value
Type	Bug
Priority	Unset
Affects Version/s	None
Labels	None
Geck Link	Create Geck

The 'Description' section contains the following text:

Error: pointer is outside its bounds  
Found in C:\Polyspace\Proj\_Zen\sources\example.c.  
Go to Polyspace finding here: <http://localhost:9443/metrics/index.html?a=review&p=5&r=5&fid=1181>



Dara is a software developer  
She is responsible for writing code and fixing defects

Browser address bar: [UXVIZ-623] Illegally dereference x  
https://jira-test-aws.mathworks.com/browse/UXVIZ-623

MathWorks Jira Dashboards Projects Issues Boards Structure MathWorks Applications Create Search

Visual Design

QE-IAT Kanban board Releases Reports Issues Components Add-ons

Visual Design / UXVIZ-623

### Illegally dereferenced pointer

Edit Comment Assign More In Progress Done To Verify

**Details**

Type:	Bug	Status:	TO DO (View Workflow)
Priority:	Unset	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	None
Labels:	None		
Geck Link:	Create Geck		

**Description**

Error: pointer is outside its bounds

Found in C:\Polyspace\Proj\_Zen\sources\example.c.

Go to Polyspace finding here: <http://localhost:9443/metrics/index.html?a=review&p=6&r=7&fid=3949> Click to edit

Critical run-time error, needs investigation.

**Attachments**

Drop files to attach, or browse.

**People**

Assignee: Unassigned  
Assign to me

Reporter: Jay Abraham

Watchers: 1 Stop watching this issue

**Dates**

Created: 2 hours ago  
Updated: 2 hours ago

**Agile**

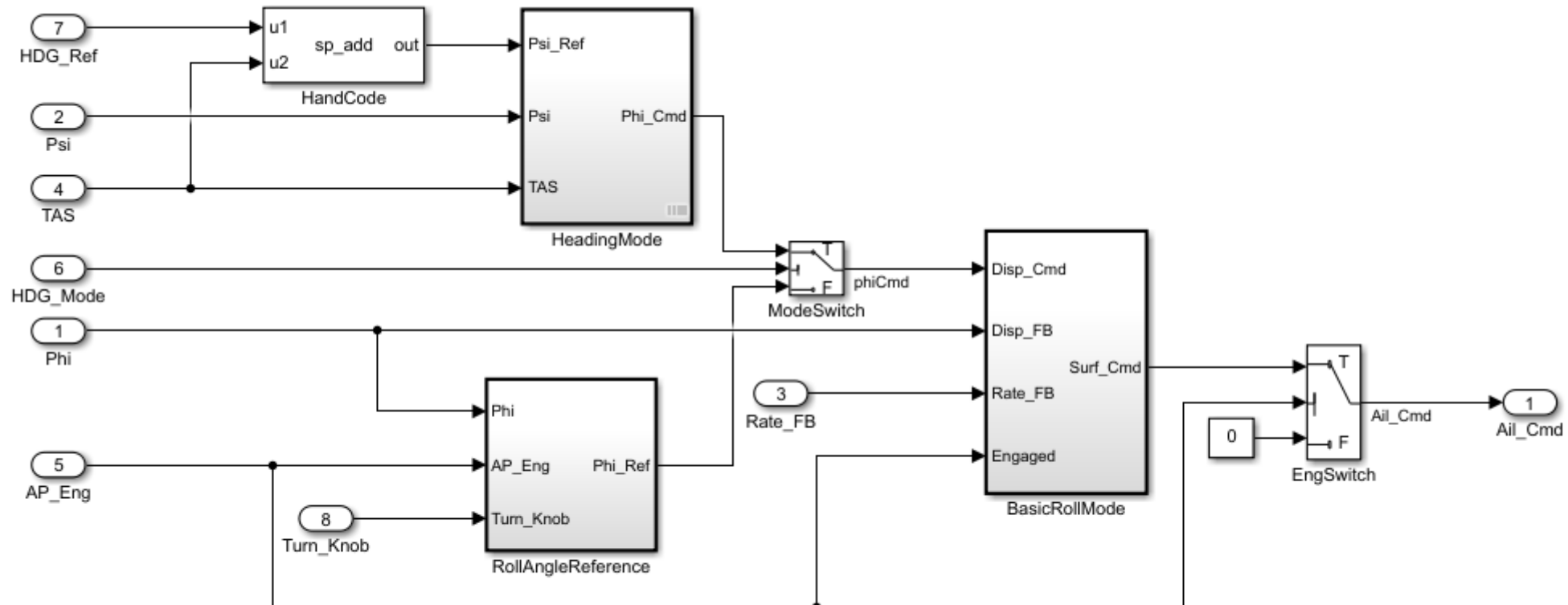
View on Board

localhost:9443/metrics/index.html?a=review&p=6&r=7&fid=3949

Get Help / Give Feedback



Eric is a Simulink and Embedded Coder user  
He is responsible for generating code from models



Copyright 1990-2018 The MathWorks, Inc.



Eric is a Simulink and Embedded Coder user  
He is responsible for generating code from models

Code Generation Report

Find:  Match Case

Contents

[Summary](#)  
[Subsystem Report](#)  
[Code Interface Report](#)  
[Traceability Report](#)  
[Static Code Metrics Report](#)  
[Code Replacements Report](#)  
[Coder Assumptions](#)

Generated Code

[-] Main file

[ert\\_main.c](#)

[-] Model files

[rtwdemo\\_roll.c](#)  
[rtwdemo\\_roll.h](#)

[+] Shared files (1)

[+] Other files (1)

Code Generation Report for 'rtwdemo\_roll'

Model Information

Author	The MathWorks, Inc.
Last Modified By	The MathWorks, Inc.
Model Version	1.162
Tasking Mode	SingleTasking

[Configuration settings at time of code generation](#)

Code Information

System Target File	ert.tlc
Hardware Device Type	Intel->x86-64 (Windows64)
Simulink Coder Version	9.1 (R2019a) 23-Nov-2018
Timestamp of Generated Source Code	Wed Apr 10 10:36:32 2019
Location of Generated Source Code	C:\Work\MATLAB\ML_Expo\rtwdemo_roll_ert_rtw\
Type of Build	Model
Memory	Global Memory: 39(bytes) Maximum Stack:

rtwdemo\_roll - Simulink

File Edit View Display Diagram Simulation Analysis Code Tools Help

rtwdemo\_roll

Copyright 1990-2018 The MathWorks, Inc.





Martin is a project manager  
He is responsible for software quality of the project

Polyspace

localhost:9443/metrics/index.html?a=metrics&p=1

DASHBOARD

Project Overview Run-time Checks Code Metrics Custom Rules MISRA C:2012

Layout Open in Desktop Review

PROJECT EXPLORER

- public
  - Proj\_Zen
  - Test\_Area

PROJECT DETAILS

Project

Name public

Tools Code Prover

Coding Standards Custom Rules, MISRA C:2012

Number of Runs 6

SUPPORT REPORT

Project Overview

Summary

Open Issues

Open	104
New	0
Assigned To Me	0
Unassigned	104

Code Metrics

Sub-project(s)	2
Number of Files	7
Number of Lines Without Comment	450
Cyclomatic Complexity	6

Run-time Checks

Selectivity 90%

Red	4
Orange	21
Gray	8
Green	300

Coding Standards

Density 151

To Do 68

# Summary

- Use Polyspace to achieve high quality software with reduced testing effort
  - Prove that your code will not cause safety hazards or security issues
- Polyspace fits software development workflows
  - Jenkins for build automation and Jira for bug tracking
- Supports team based collaboration
  - Results published for web-browser based review by developers and quality engineers
  - Dashboards to show quality metrics for project and safety managers.



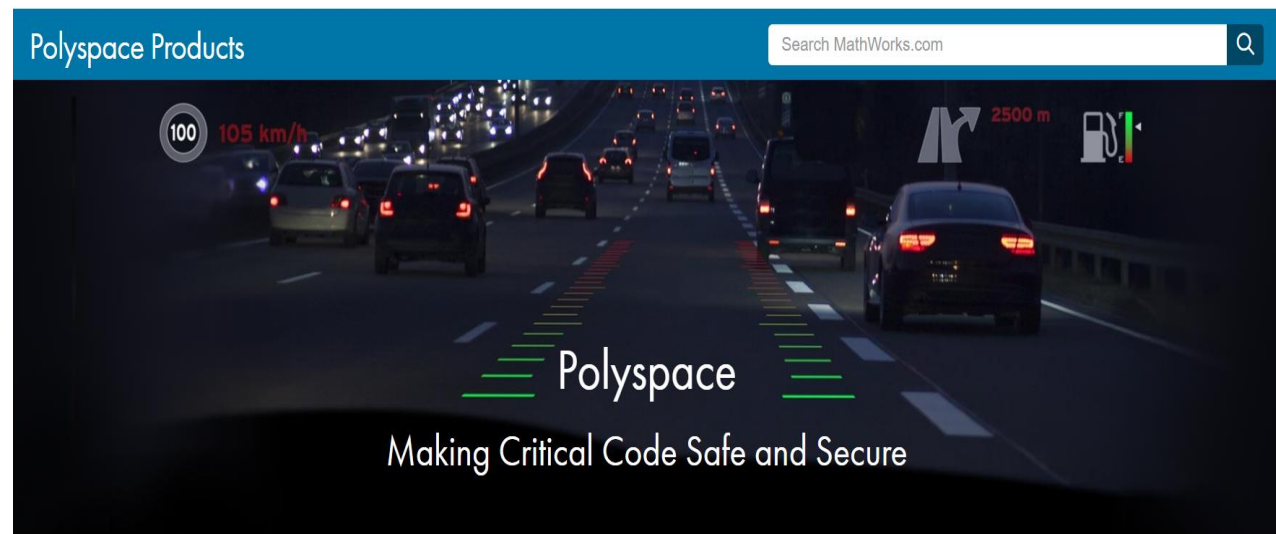
# Polyspace Helps Makes C, C++, and Ada Safe and Secure

Safety		Security	
<b>Standards:</b> <ul style="list-style-type: none"><li>• DO-178 (aero)</li><li>• ISO 26262 (auto)</li><li>• IEC 61508 (industrial)</li><li>• IEC 62304 (medical)</li><li>• EN 50128 (rail)</li></ul>		<b>Standards:</b> <ul style="list-style-type: none"><li>• MISRA</li><li>• AUTOSAR</li></ul>	
		<ul style="list-style-type: none"><li>• CERT-C</li><li>• CWE</li><li>• ISO 17961</li><li>• MISRA-C:2012 Appendix 1</li><li>• Tainted data tracking</li></ul>	
Reliability and Robustness			
<b>Code Proving</b> <ul style="list-style-type: none"><li>• Prove absence of critical runtime errors (or find even the slightest vulnerability)</li><li>• Exhaustive: all possible inputs, control flows, data flows (no instrumentation, execution, test cases)</li><li>• Sound: no false negatives</li></ul>			
Quality			
<ul style="list-style-type: none"><li>• Coding Standards</li><li>• Find Probable Bugs, Defects</li><li>• Code Metrics</li><li>• Formal Method: Runtime Behavior, Debugger-like view</li><li>• Review Scopes / Software Quality Objectives</li><li>• Simulink Integration: trace issues in generated code back to model</li></ul>			

## Learn More

Visit MathWorks Code Verification Solution Page:

<https://www.mathworks.com/products/polyspace.html>



Polyspace® static code analysis products use formal methods to prove the absence of critical run-time errors under all possible control flows and data flows. They include checkers for coding rules, security vulnerabilities, code metrics, and hundreds of additional classes of bugs.



## Polyspace for C/C++ Code Verification

This two-day course discusses the use of Polyspace Bug Finder™ and Polyspace Code Prover™ to prove code correctness, improve software quality metrics, and ensure product integrity.

### Topics include:

- Creating a verification project
- Reviewing and understanding verification results
- Emulating target execution environments
- Handling missing functions and data
- Managing unproven code (color-coded in orange by Polyspace® products)
- Applying MISRA C® rules
- Reporting

# Thank You