

MATLAB EXPO 2019

Making Software Safe and Secure
with Team Collaboration

Polyspace Presentation for MATLAB Expo



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()

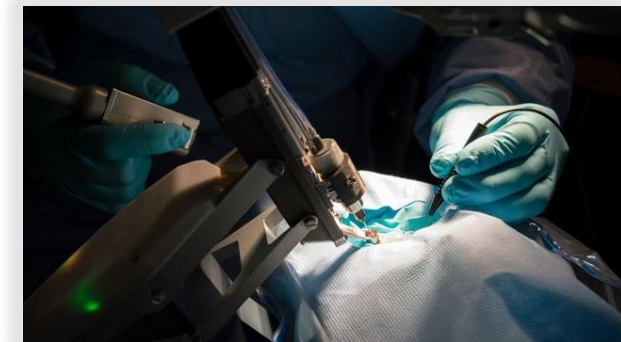
● Non-terminating loop ?

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 shows the Polyspace web interface. At the top, there are browser tabs for 'Polyspace', 'Polyspace Access Cluster Operat...', and 'System Dashboard - Jira'. The address bar shows the URL 'localhost:9443/metrics/index.html?a=review&p=3&r=2'. Below the browser, the Polyspace interface has a navigation bar with tabs like 'Dashboard', 'Run-time Checks', 'Defects', 'Coding Standards', 'Code Metrics', and 'Global Variables'. A 'Filters' section is visible with options like 'Filter out', 'Comment, filename, etc.', 'Layout', and 'Open in Desktop'. The main content area shows a 'Results List' table with columns for ID, Type, Group, Check, and Information. The table lists several items, with ID 117 highlighted as a 'Green Check' for 'Division by zero'. Below the table, there is a 'Result Details' section with fields for 'Status' (Unreviewed), 'Severity' (Unset), and 'Assigned to'. A 'Track issue' button is also present. On the right side, a 'Source Code View' shows a C code snippet with a division operation. Blue callout boxes with arrows point to various parts of the interface: 'Defect Details' points to the 'Result Details' section, 'Filter Results' points to the 'Filters' section, 'Defect List' points to the table, and 'Source Code View' points to the code editor.

ID	Type	Group	Check	Information
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	-

```

7  k = input
8  x = 2.0f;
9  y = k + 5.0f;
10
11 while (x < 10.0f)
12
13     where_are_the_errors_float(float input)
14
15
16
17     return (-9999.0f);
18
19 y++;
20 x = x / (x - y);
21 }
22
23 return x;
24 }
25
26
    
```

Defect Details

Filter Results

Defect List

Source Code View

Where_Are_Errors_Float2

Polyspace Tools

Bug Finder

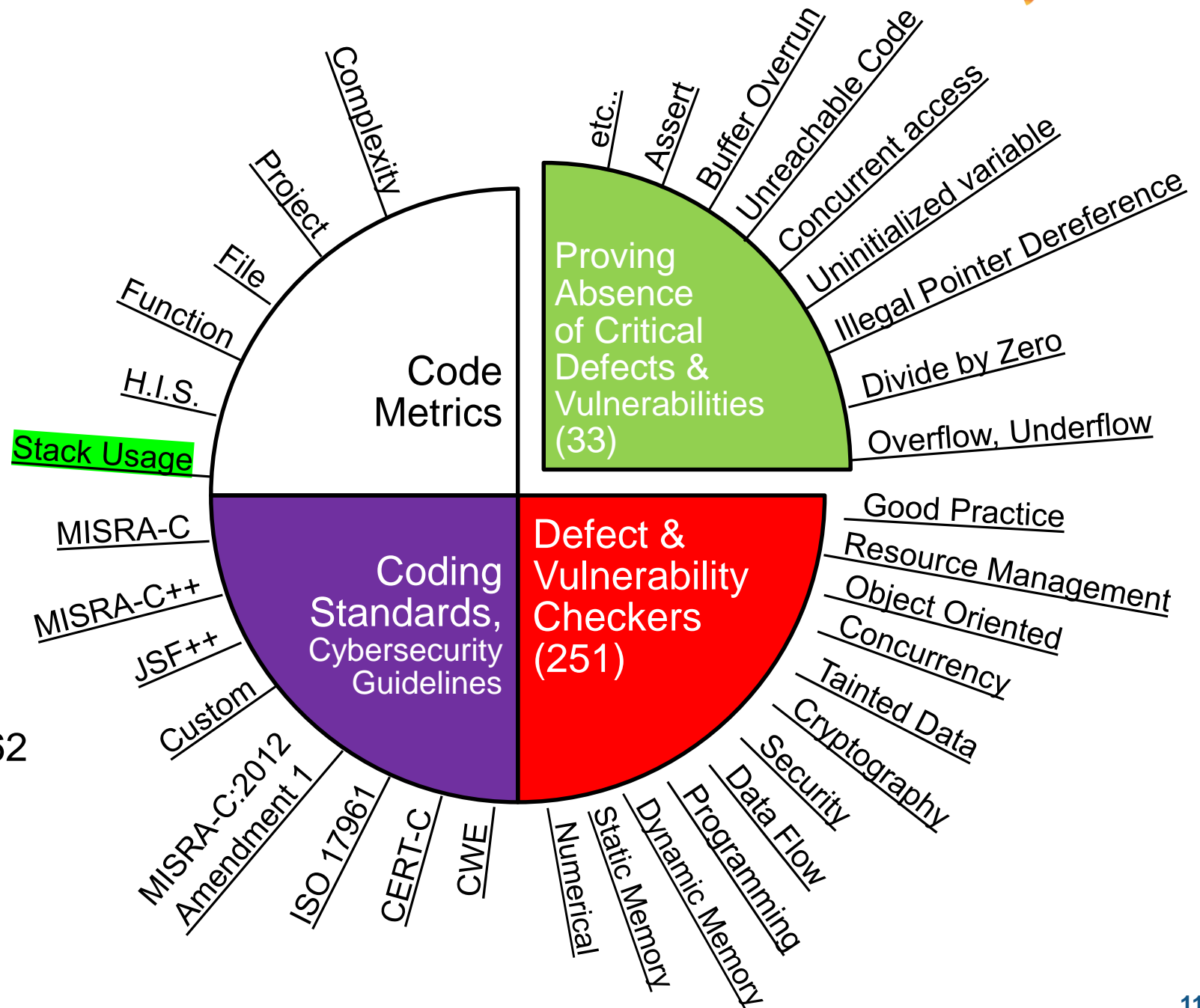


- Produce code metrics
- Check coding standards
- Find defects and vulnerabilities

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

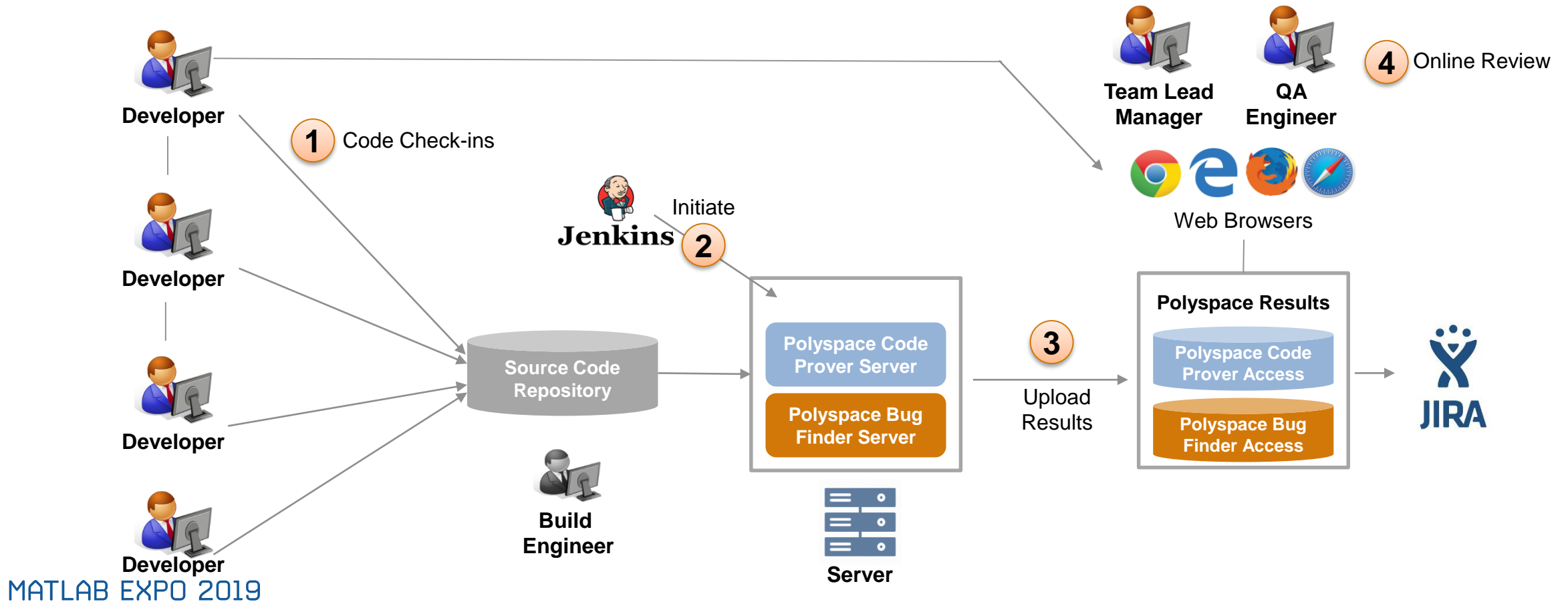


Miracor Eliminates Run-Time Errors and Reduces Testing Time for Class III Medical Device Software

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 | 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 02:41:1
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

PROVISION
 START ALL
 STOP ALL
 DELETE ALL

Service	Status	Action
User Manager	Running	Stop
Database	Running	Stop
ETL	Running	Stop
Web Server	Running	Stop
Gateway	Running	Stop

[Services](#) | [Nodes](#) | [Settings](#)



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

Polyspace Code Verification: 114 new findings for project...

File Message Help Mimecast Tell me what you want to do

Sun 3/17/2019 6:02 PM

Bob Builder

Polyspace Code Verification: 114 new findings for project Zen

To: Quin Quality

mail_details.html
62 KB

Polyspace found 114 new findings when analyzing 'xent':

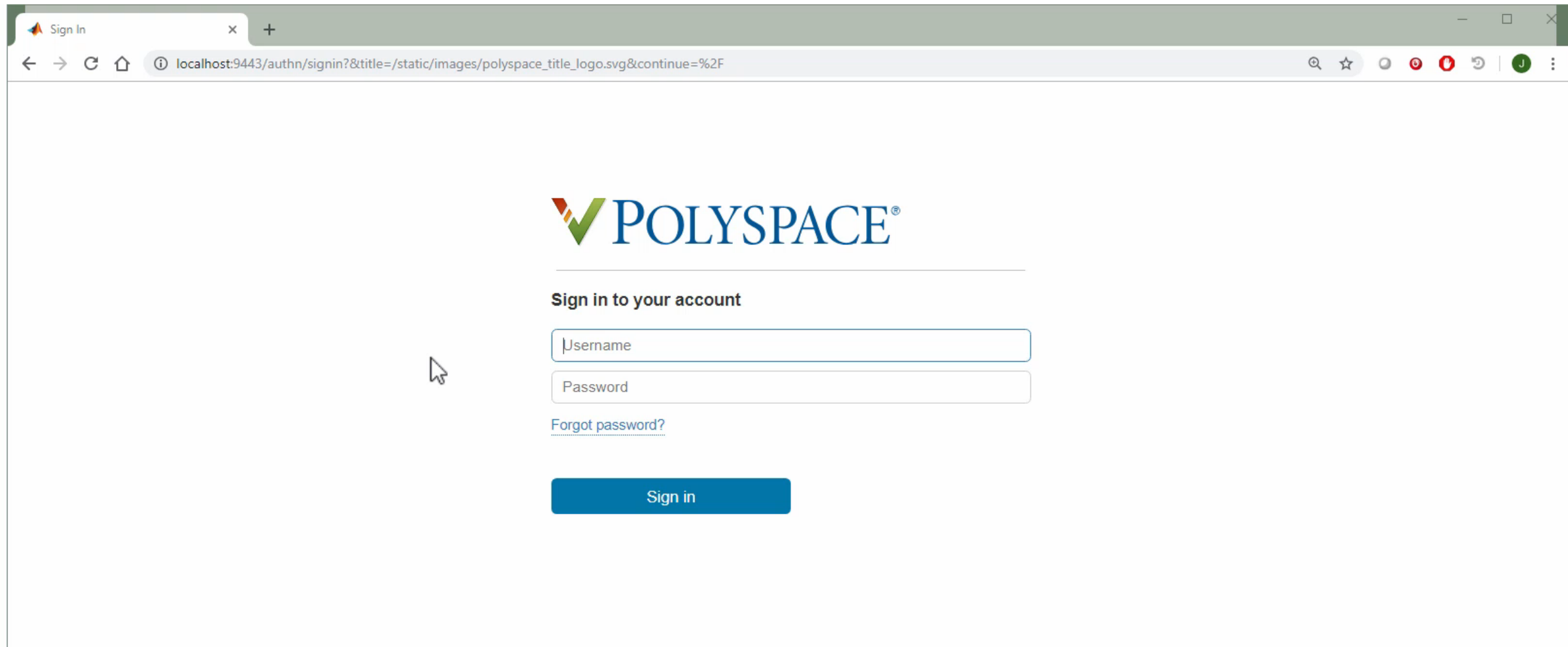
- To view details, check attached file and follow urls.
- To go to directly to project, follow: <https://polyspace-access:9443/metrics/index.html?a=review&p=81&r=1898>.

You can see the Jenkins log file here: http://jenkins-polyspace:8080/job/polyspace_modules/38/console.

Bob Builder
Build Engineer, Tools Group
(508) 647-3027 bbuilder@mathworks.com



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



The screenshot shows a web browser window with the following elements:

- Browser tab: "Sign In"
- Address bar: "localhost:9443/authn/signin?&title=/static/images/polyspace_title_logo.svg&continue=%2F"
- Page content:
 - Polyspace logo (a green checkmark icon followed by the text "POLYSPACE®")
 - Section header: "Sign in to your account"
 - Form fields: "Username" and "Password" (the password field is currently empty)
 - Link: "[Forgot password?](#)"
 - Button: "Sign in" (a blue button)



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 Jira ticket in Project Zen. The ticket is titled "Illegally dereferenced pointer" and is a Bug. The description includes an error message and a link to the Polyspace finding.

Project Zen

Visual Design / UXVIZ-620

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=5&r=5&fid=1181>

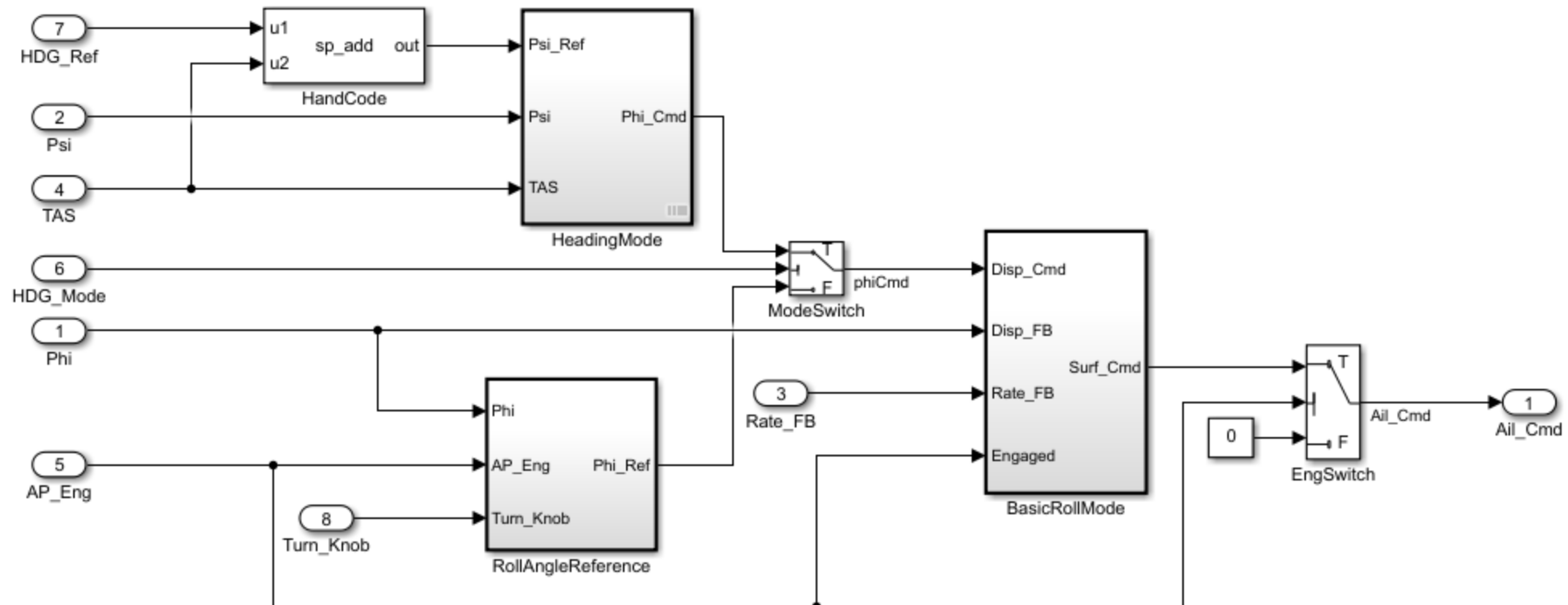


Dara is a software developer
She is responsible for writing code and fixing defects

The screenshot shows a Jira issue page for 'Illegally dereferenced pointer' in the 'Visual Design' project. The issue is currently in the 'In Progress' state. The details section shows it is a 'Bug' with an 'Unset' priority and 'Unresolved' status. The description contains an error message: 'Error: pointer is outside its bounds' found in 'C:\Polyspace\Proj_Zen\sources\example.c'. A link to a local host is provided: <http://localhost:9443/metrics/index.html?a=review&p=6&r=7&fid=3949>. A tooltip 'Click to edit' is visible over the link. The 'People' section shows the issue is 'Unassigned' and reported by 'Jay Abraham'. The 'Dates' section shows it was created and updated '2 hours ago'. The 'Attachments' section is empty with a 'Drop files to attach, or browse.' prompt.



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 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	ert.tlc
File	
Hardware	Intel->x86-64 (Windows64)
Device Type	
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:

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

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 (Open: 32)

Selectivity 90%

Red	4
Orange	21
Gray	8
Green	300

Coding Standards (Open: 68)

Density 151

- To Do: 68

public

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.

End