

A graphic of the MATLAB logo, consisting of five overlapping triangles in shades of blue and orange, positioned on the left side of the image.

MATLAB EXPO 2018  
KOREA

# MATLAB EXPO 2018

## Reducing Testing Effort with Static Code Verification

테스팅 비용 감소를 위한 정적 코드 검증 활용 방안

유용출 과장 (gary.ryu@mathworks.com)



# Agenda

- Misconceptions lead Testing effort high
- Polyspace helps you to reduce testing effort
- Conclusion

## Misconceptions lead Testing Efforts High

1. Our expert programmers don't make mistakes
2. I need Dynamic Testing only, I do not need Static Analysis
3. I do Static Analysis as the last quality gate before release

# 1. Our expert programmers don't make mistakes

- They are infallible! Aren't they?
- Look at this table,

Group	Average Bugs per KLOC
Top 1% developers	11.2
Top 10% developers	28.9
Top 25 % developers	61.9
All	120.8

Defect Injection Ranges for 810 Experienced Software Developers  
 The Ganssle Group, Derived from articles written by Watts Humphrey(Father of Software Quality)

# 1. Our expert programmers don't make mistakes

- They are infallible! Aren't they?

- Look

**Finding Bugs with Right Tool  
is Must-Do-activity!**

Top 25 % developers	61.9
All	120.8

Defect Injection Ranges for 810 Experienced Software Developers  
The Ganssle Group, Derived from articles written by Watts Humphrey(Father of Software Quality)

## 2. I need Dynamic Testing only, I do not need Static Analysis

- I do sufficient testing: Unit testing, Integration testing, Field testing ...
- It's redundant to do both of dynamic testing and static analysis

### DEFECT REMOVAL EFFICIENCY CASE 4 (Worst)

No Inspections; No static analysis. **Testing Only**

#### DEVELOPMENT DEFECTS REMOVED

Static analysis	0
Inspections	0
Testing	850
Subtotal	850

#### USER-REPORTED DEFECTS IN FIRST 90 DAYS

Valid unique defects **150**

#### TOTAL DEFECT VOLUME

Defect totals 1,000

#### DEFECT REMOVAL EFFICIENCY

Dev. (850) / Total (1,000) = 85.0%

Copyright © 2012-2015 by Capers Jones. All Rights Reserved.

### DEFECT REMOVAL EFFICIENCY CASE 1 (BEST)

**Inspections + static analysis + testing**

#### DEVELOPMENT DEFECTS REMOVED

Static analysis	350
Inspections	390
Testing	250
Subtotal	990

#### USER-REPORTED DEFECTS IN FIRST 90 DAYS

Valid unique defects **10**

#### TOTAL DEFECT VOLUME

Defect totals 1,000

#### DEFECT REMOVAL EFFICIENCY

Dev. (990) / Total (1,000) = 99.0%

Copyright © 2012-2015 by Capers Jones. All Rights Reserved.

# Quiz: How many tests to achieve 100% MCDC

```

1  #include "simple_lookup_table.h"
2
3  const double x[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
4  const double y[10] = {3, 6, 9, 12, 15, 18, 21, 24, 27, 30};
5
6  double simple_lookup_tbl (double in1, sint8_t * stat)
7  {
8      double out;
9      double f;
10     uint32_t i;
11
12     if (in1 < x[0])
13     {
14         /* lower saturation */
15         out = y[0];
16         *stat = -1;
17     }
18     else if (in1 > x[9])
19     {
20         /* upper saturation */
21         out = y[9];
22         *stat = -1;

```

**Answer: 6 tests**

```

23     }
24     else
25     {
26         for (i = 4; in1 < x[i]; i--);
27
28         if (i > 2 && in1 > 5.0F)
29         {
30             1.0F;
31
32             while (in1 >= x[i + 10]) {
33                 i++;
34             }
35
36             f = (in1 - x[i]) / (x[i + 10] - x[i] + 0.000000001F);
37
38             out = (y[i + 10] - y[i]) * f + y[i];
39             *stat = 0;
40         }
41     }
42
43     return out;
44 }

```

- Automatically-generated test for input = 5.9566, 1.9184, -5.4067, 9.8142, 11, 4

## Let's see the coverage report of the hand-written code

**Don't we miss something dangerous?**

**S-Function Code Coverage Results:**

Complexity	Decision	Condition	MCDC	Statement
<b>TOTAL COVERAGE</b>	100%	100%	100%	100%
1... <a href="#">sfunc_lookup_tbl</a> 6	100%	100%	100%	100%

# Runtime error lurking in the code

Improve SW quality  
with Polyspace!

? Out of bounds array index

Warning: array index outside bounds: [0, 9]  
This  
arra  
arra

1  
2  
3  
4

S

sim

27 for (i = 4; in1 < x[i], i--),

28 if (i > 2 && in1 > 5.0F) {

30 f = 1.0F;

31 }

32

33 while (in1 >= x[i + 10]) {

34 i++;

35 }

36

37 f = (in1 - x[i]) / (x[i + 10] - x[i] + 0.0000000001F);

**Polyspace Code Prover finds this runtime error**

- Array x allocated for 10 elements
- Range of index is 1 .. 10
- **Potential to access  $x[10]$ , out of bounds array**

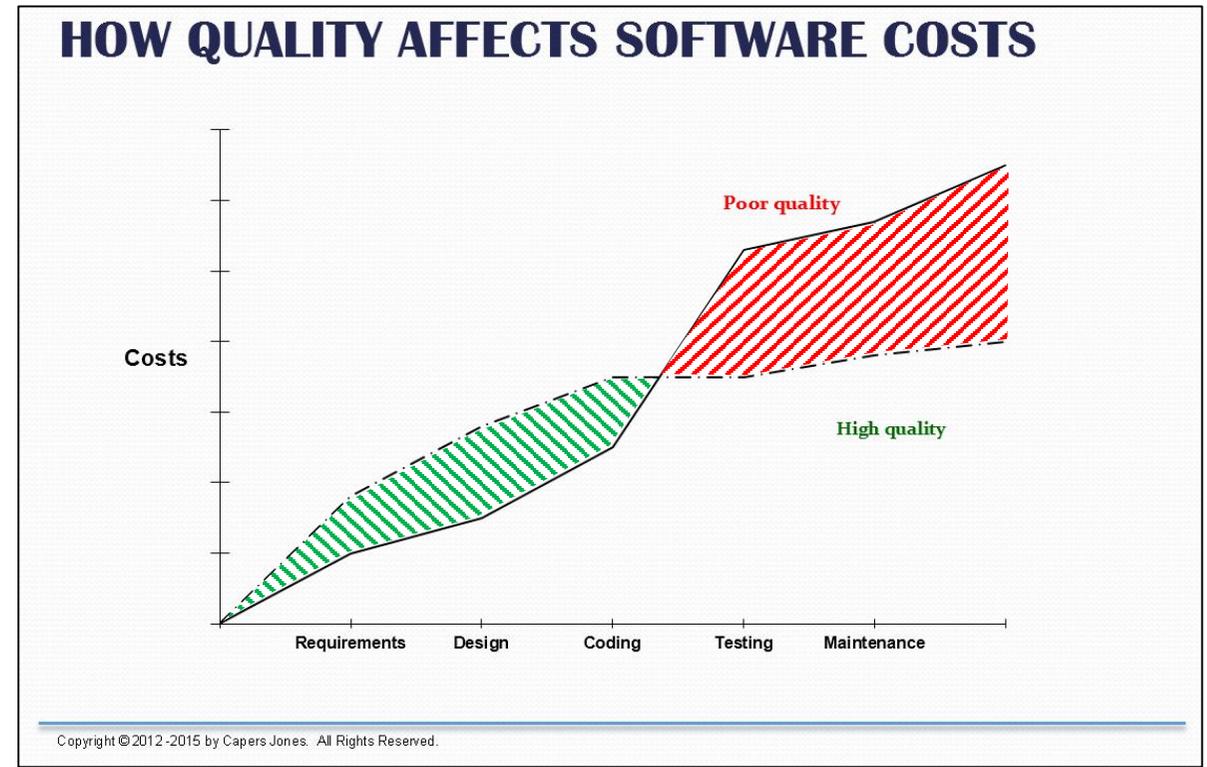
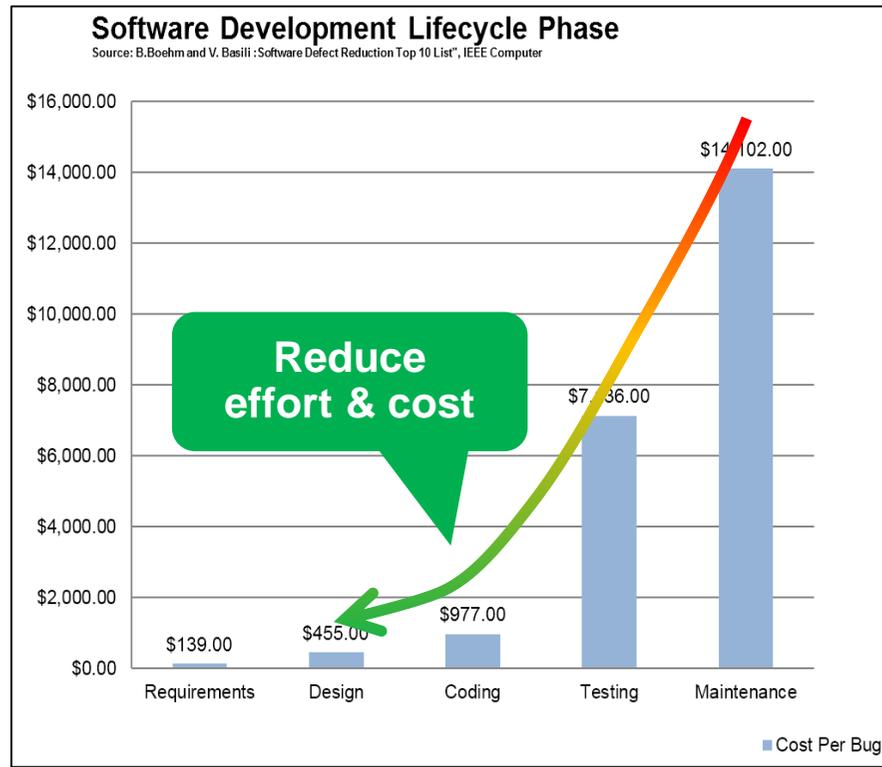
Element of global array (float 64): integer values in [1.0 .. 10.0]

array size: 10

array index value: [1 .. 10]

### 3. I do Static Analysis as the last quality gate before release

- Efficient Defect Reduction
  - Cost less in early stage
- Accumulated Technical Debt
  - Too many things to review or fix



### 3. I do Static Analysis as the last quality gate before release

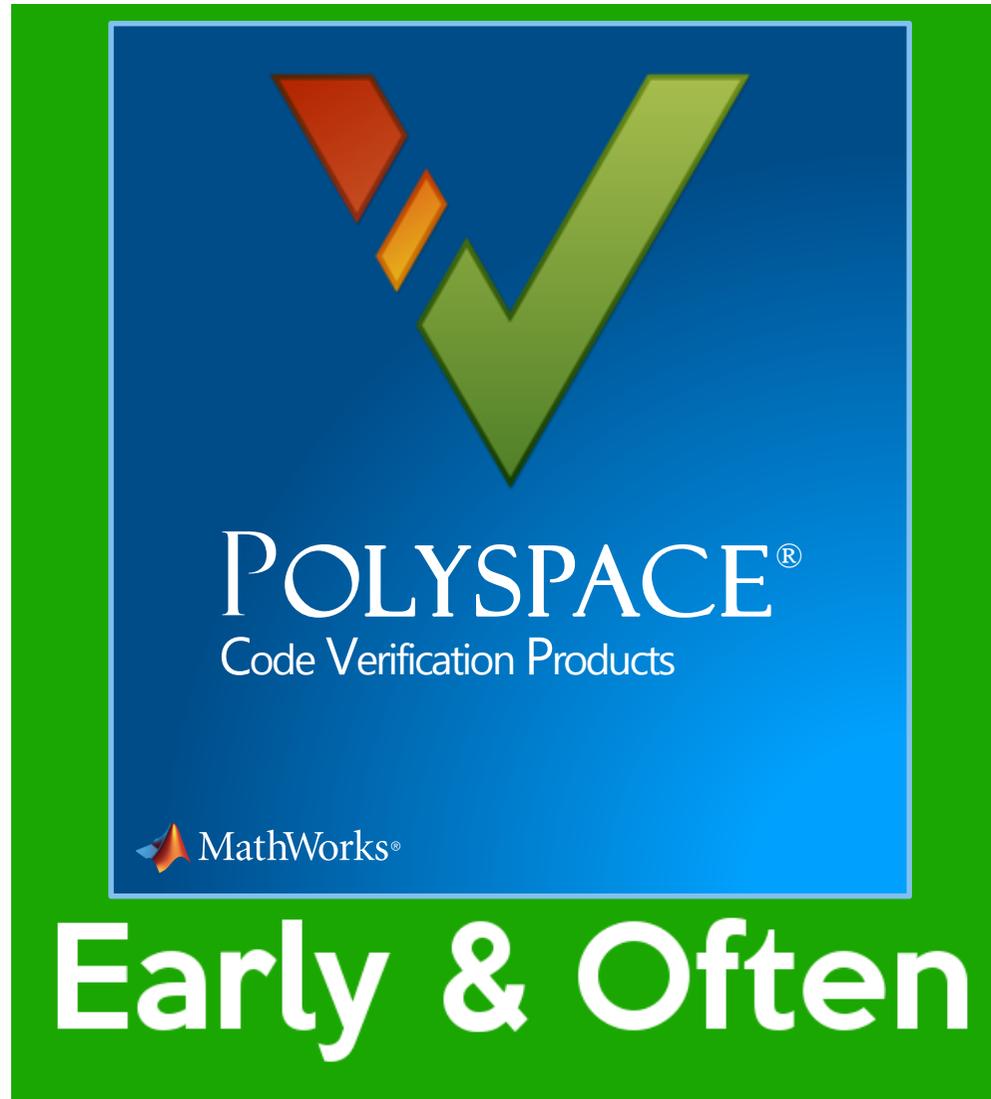
- Efficient Defect Reduction
  - Cost less in early stage
- Accumulated Technical Debt
  - Too many things to review or fix



# Lessons learned from misconceptions

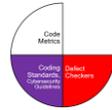


# Lessons learned from misconceptions



# Polyspace helps you to ...

## Bug Finder



→ High Quality, Secure, Compliant Code:

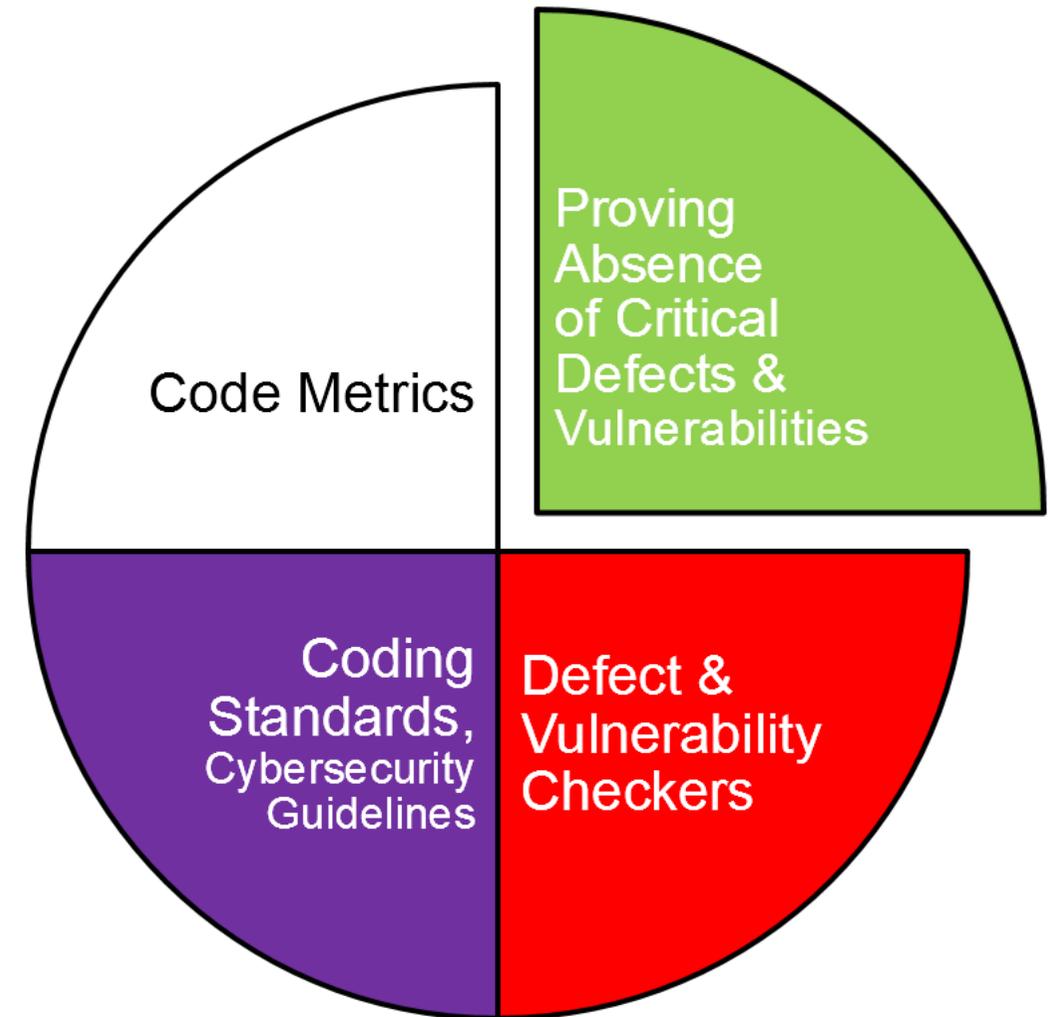
- Various defects or vulnerabilities
- Credits for functional safety, cybersecurity standards.

## Code Prover



→ Fully Trusted Components:

- Proven free of critical runtime defects and vulnerabilities
- Additional credits for standards.



# Save-time analysis workflow in Eclipse

Polyspace - ctrl\_lib\_example/Sources/mylibrary.c - CodeWarrior Development Studio

File Edit Source Refactor Navigate Search Project MQX Tools Processor Expert Polyspace Run Window Help

Project Explorer

- ctrl\_lib\_example
  - Archives
  - Includes
  - Debug
  - Project Headers
    - mylibrary.h
  - Sources
    - mylibrary.c

Results List - Bug Finder Fast

All results Showing 29/29

Family	Group	Check
mylibrary.c	Defects: 7 - Coding Rules: 22	
ctrl_sbr()	Defects: 5 - Coding Rules: 11	
	Programming	Invalid use of = operator
	Static memory	Unreliable cast of pointer
	Good practice	Missing break of switch case
	Good practice	Unused parameter
	Data flow	Variable shadowing
	10 The essential type model	10.1 Operands shall not be
	16 Switch statements	16.1 All switch statement
	2 Unused code	2.7 There should be no u
	11 Pointer type conversions	11.3 A cast shall not be p
	16 Switch statements	16.4 Every switch statem
	16 Switch statements	16.3 An unconditional bre
	5 Identifiers	5.3 An identifier declared
	10 The essential type model	10.1 Operands shall not b
	10 The essential type model	10.4 Both operands of an
	10 The essential type model	10.1 Operands shall not b
	13 Side effects	13.4 The result of an assi
	File Scope - Coding Rules: 4	
	lpf_speed() - Defects: 2 - Coding Rules: 7	

```

mylibrary.c
mylibrary.h
+* This is a hand code to implement Seatbelt reminder algorithm.

#include "mylibrary.h"

float gPrev_speed;

static sint32_t lpf_speed (sint32_t * output_Filtered_speed, float * input_curr_speed)
(
    sint32_t ERRORSTATUS;
    float tmp_filtered_speed;

    tmp_filtered_speed = (0.1F * (*input_curr_speed)) + (0.9F * (gPrev_speed));

    if (tmp_filtered_speed >= 0.5F) {
        if (tmp_filtered_speed >= 1000.00F) {
            ERRORSTATUS = ERROR_CTRL;
        } else {
            *output_Filtered_speed = (sint32_t) (tmp_filtered_speed);
            ERRORSTATUS = NOERROR_CTRL;
            gPrev_speed = tmp_filtered_speed;
        }
    }
    return ERRORSTATUS;
} else {

```

Result Details

Variable trace mylibrary.c / ctrl\_sbr()

64 ERRORSTATUS = lpf\_speed(&Filtered\_speed, input\_

Result Review

MISRA C:2012 11.3 (Required) ?

A cast shall not be performed between a pointer to object type and a pointer to a different object type.

Problems Console Polyspace Run - Bug Finder

Output Summary Run Log

Total elapsed time: 00:00:02 Analysis process completed

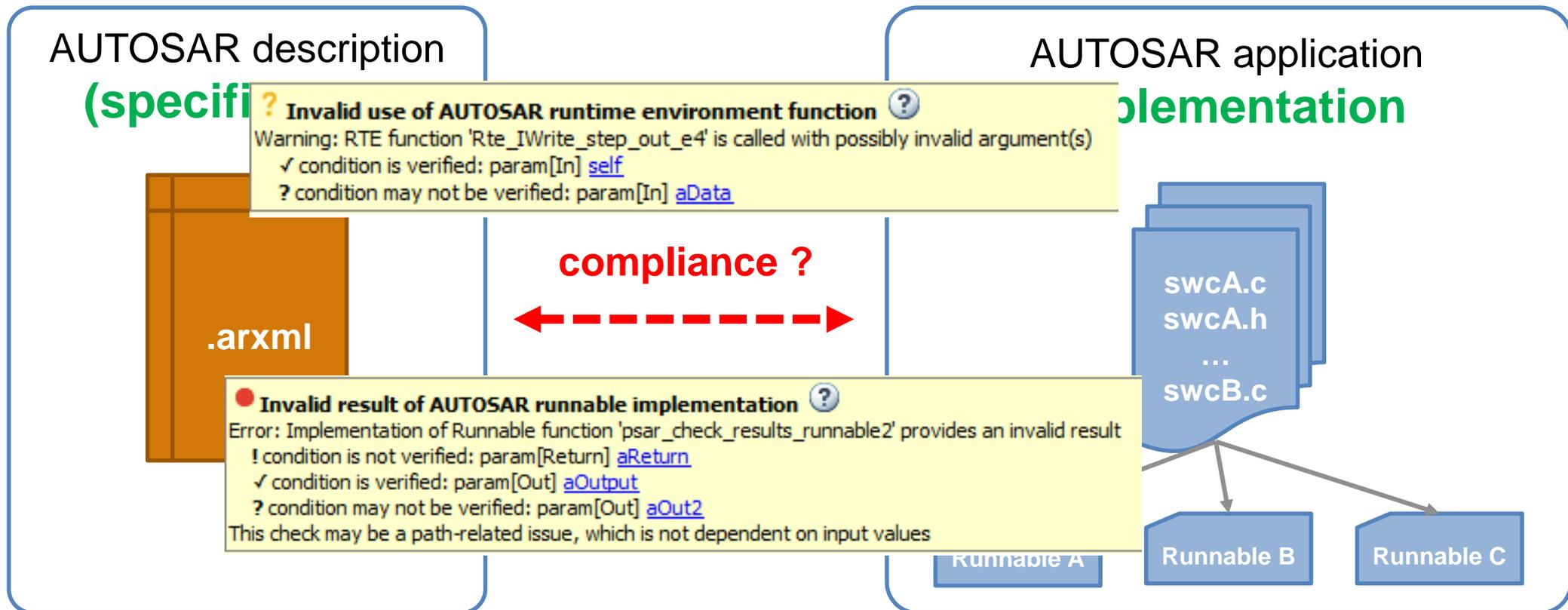
Type	Message	File	Line	Col
	C analysis starts at Mon Apr 09 13:18:56 2018			
	By default, some results are not generated for headers.			
	2 core(s) detected. The verification uses 2 core(s).			

Detail

Remote System Explorer Operation

# Unit-proving of AUTOSAR components

- Detect runtime errors early in AUTOSAR design process
- Detect inconsistencies between AUTOSAR specification and Source code



# Unit-proving of AUTOSAR components

ar\demo001\swc001\bhv001\CP\_Result

Result Details  
swc001.c / periodic\_runnable()  
Specification: allocated  
✓ self->Rte\_Dummy meets its specification.  
Specification: [0 .. 255]  
Actual value (unsigned int 8): full-range [0 .. 255]

• Conditions on second argument 'aData' (see [parameter spec](#)):  
? aData may not meet its specification.  
Specification: 4, 5, 9  
Actual value (int 32): [5 .. 6] or 10  
This check may be a path-related issue, which is not dependent on input values

Event  
1 Assignment to local pointer 'aData'  
2 Assignment to local variable 'e6'  
3 ? Warning: Function 'Rte\_IWrite\_periodic\_runnable\_out\_e6' is called with possibly invalid literal

AUTOSAR Behavior Specification  
Focus on: One Function Specific Parameter  
Rte\_IWrite\_periodic\_runnable\_out\_e6  
Function required by Autosar Software-Component

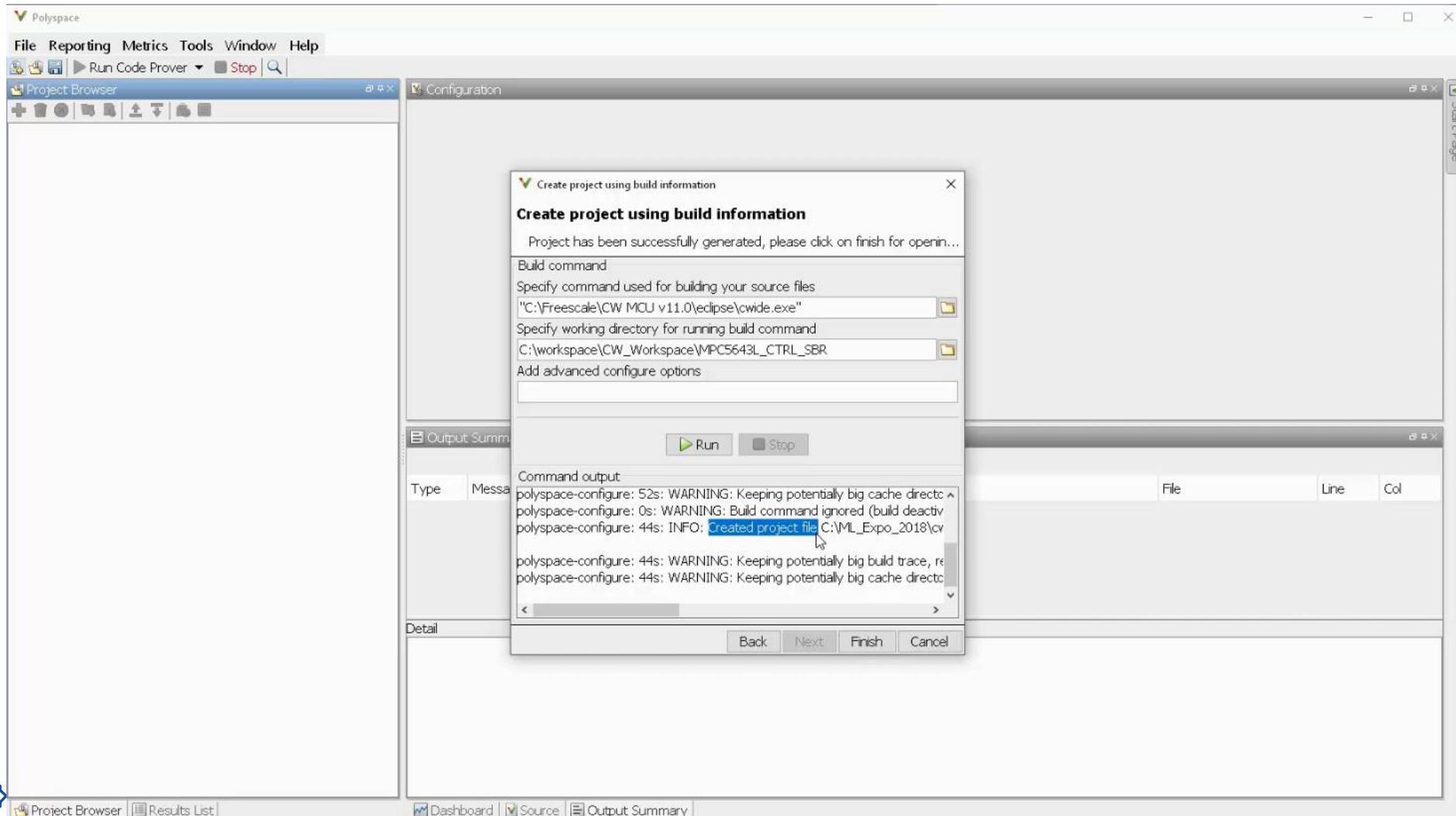
signature  
[2] IN aData is a Enumeration application type  
psar.types.app.Enum001  
Values must be one of  
• Red (4)  
• Blue (5)  
• Green (9)

project-checksum=(2336258061233110428)

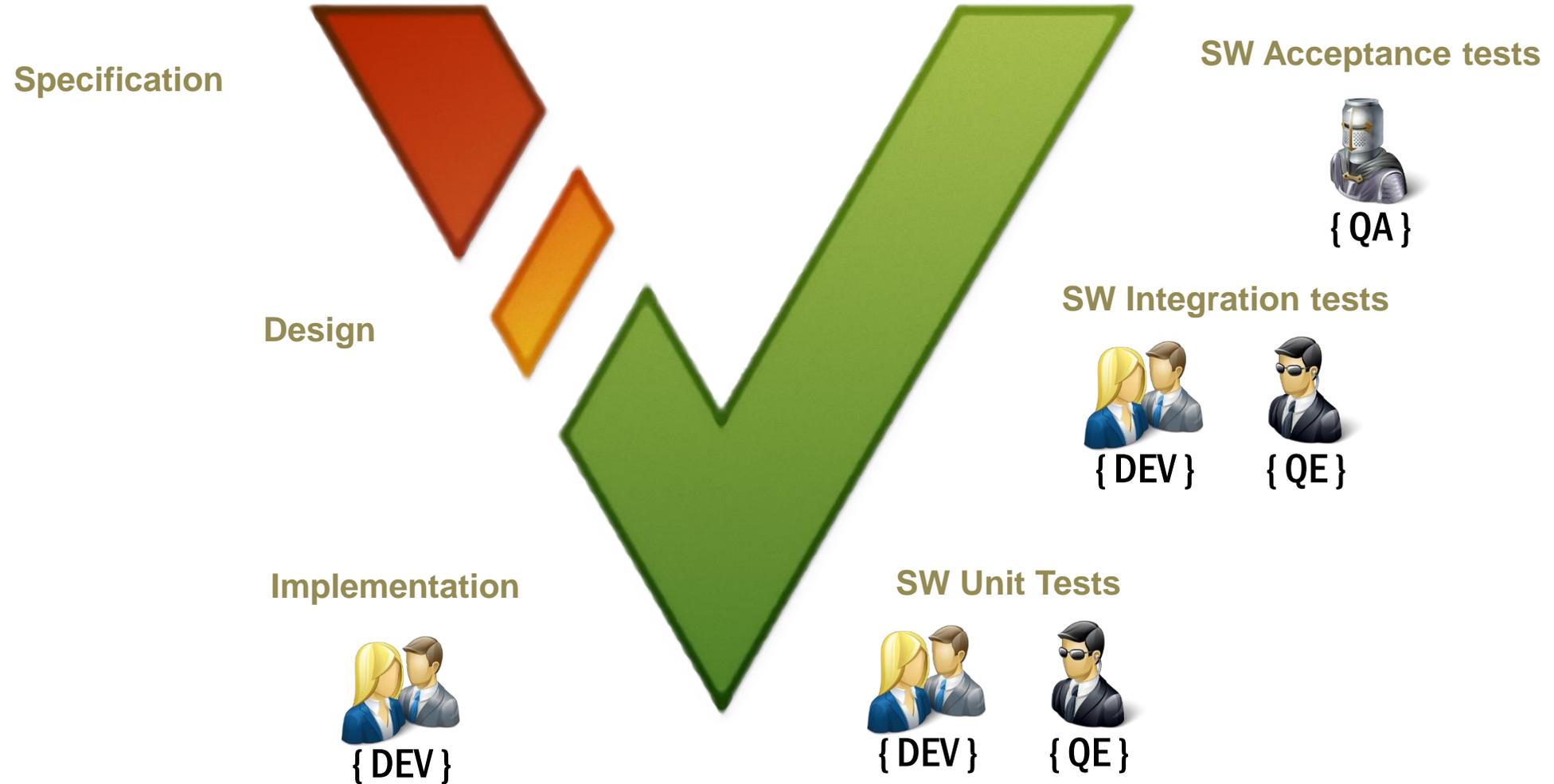
Source  
swc001.c  
88 Rte\_IWrite\_periodic\_runnable\_out\_e5(self, e5);  
89  
90 /\* read enumeration \*/  
91 app\_Enum001 e6;  
92 e6 = Rte\_IRead\_periodic\_runnable\_in\_e6(self);  
93 /\* write possibly invalid literal \*/

# Ease of configuring projects

- **Create project automatically** (DIAB, TASKING, GreenHills, IAR, CodeWarrior, TI CCS, GCC, Visual Studio)



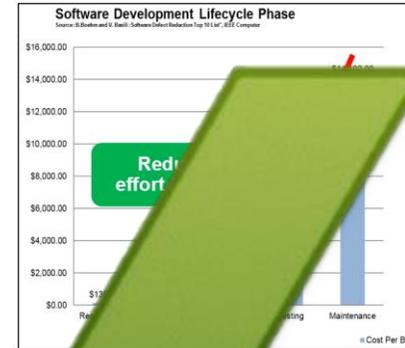
# When/Who using Polyspace products?



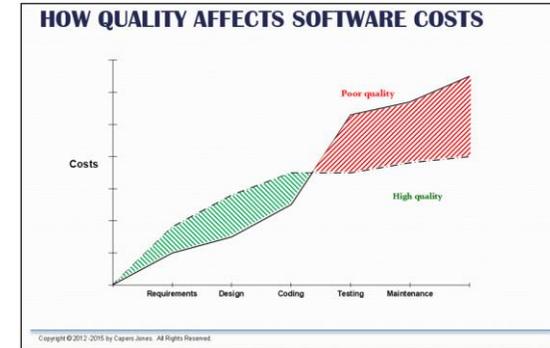
# Conclusion

Group	Avg Bugs per KLOC
Top 1% developers	11.2
Top 10% developers	28.9
Top 25 % developers	61.9
All	120.8

- Efficient Defect Reduction
  - Cost less in early stage



- Accumulated Technical Debt
  - Too many things to review or fix



  
**Verify Software**  
 with  
**Right tools**  
**Early & Often**

