
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?
- Loo

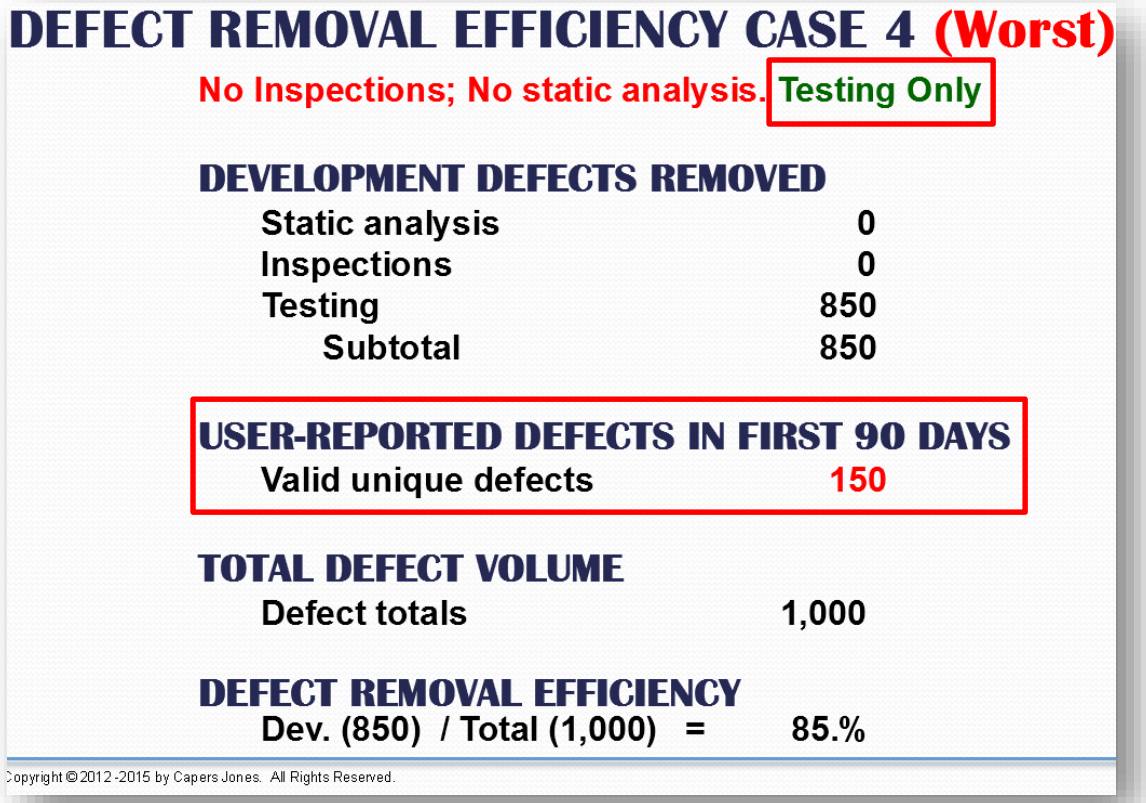
Top 25 % developers	61.9
All	120.8

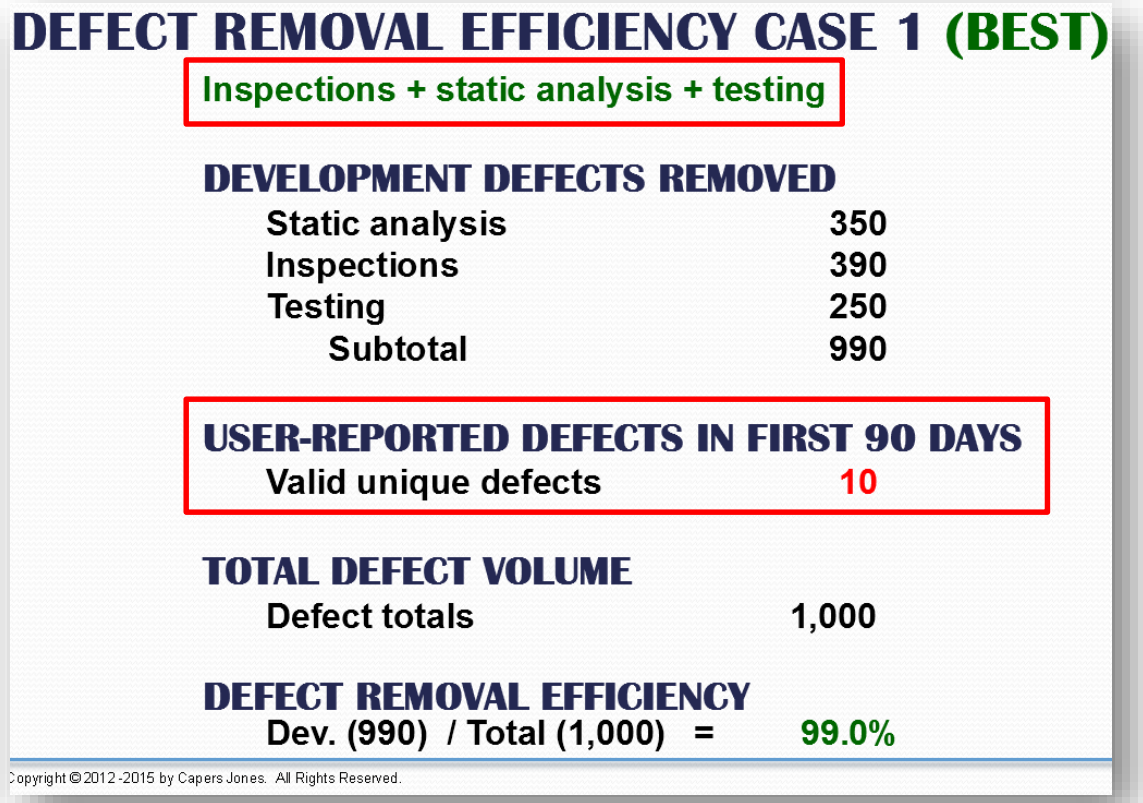
Defect Injection Ranges for 810 Experienced Software Developers
The Ganssle Group, Derived from articles written by Watts Humphrey(Father of Software Quality)

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

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





Capers Jones, "Software Quality in 2016: A Survey of the state of the art"

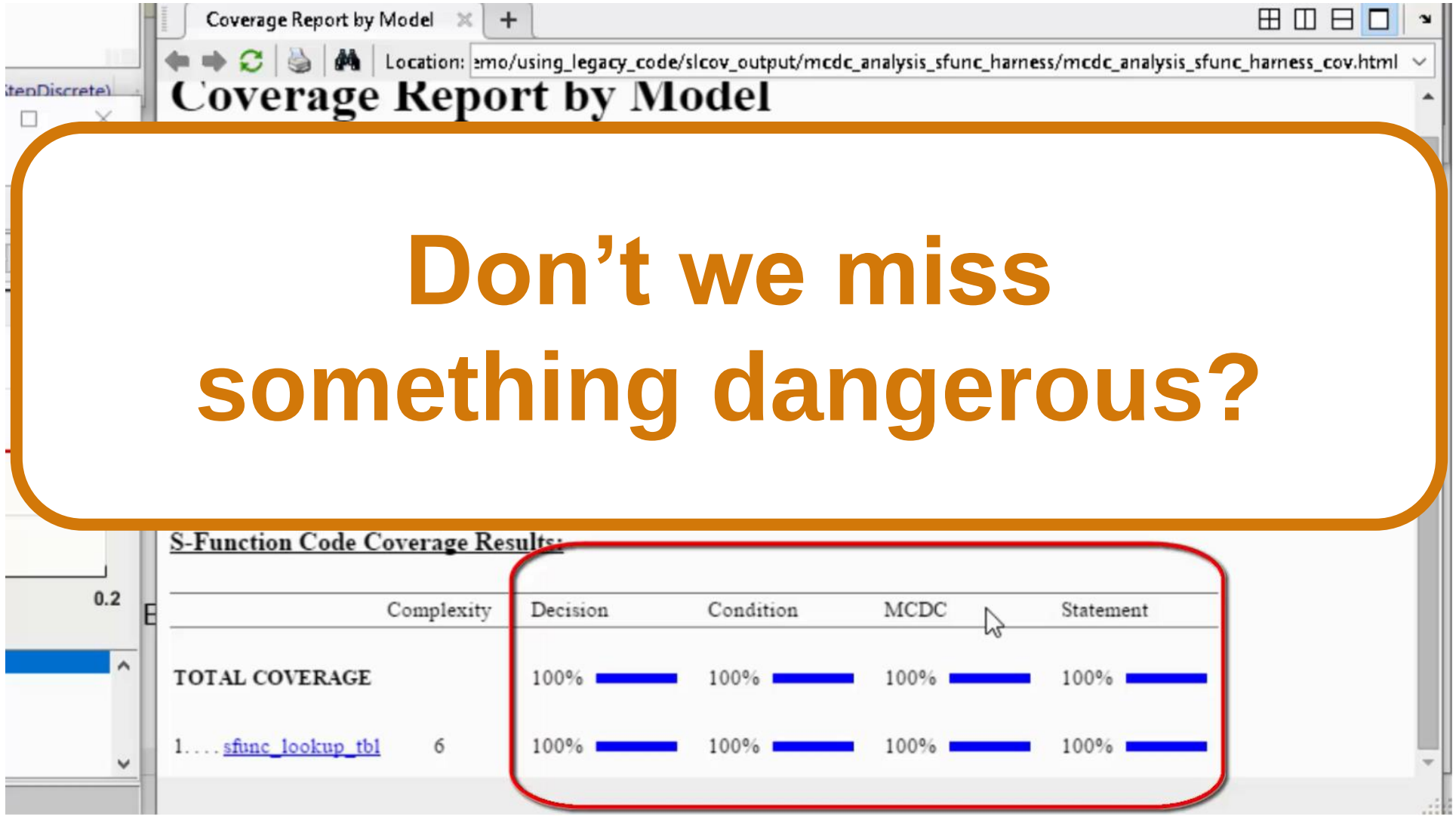
Quiz: How many tests to achieve 100% MCDC

```c
#include "simple_lookup_table.h"

const double x[10] = {1,  2,  3,  4,  5,  6,  7,  8,  9, 10};
const double y[10] = {3,  6,  9, 12, 15, 18, 21, 24, 27, 30};

double simple_lookup_tbl (double in1, sint8_t * stat)
{
    double out;
    double f;
    uint32_t i;

    if (in1 < x[0])
    {
        /* lower saturation */
        out = y[0];
        *stat = -1;
    }
    else if (in1 > x[9])
    {
        /* upper saturation */
        out = y[9];
        *stat = -1;
    }
    else
    {
        for (i = 4; in1 < x[i]; i--);

        if (i > 2 && in1 > 5.0F)
        {
            1.0F;

            while (in1 >= x[i + 1U]) {
                i++;
            }

            f = (in1 - x[i]) / (x[i + 1U] - x[i] + 0.0000000001F);

            out = (y[i + 1U] - y[i]) * f + y[i];
            *stat = 0;
        }

        return out;
}
```

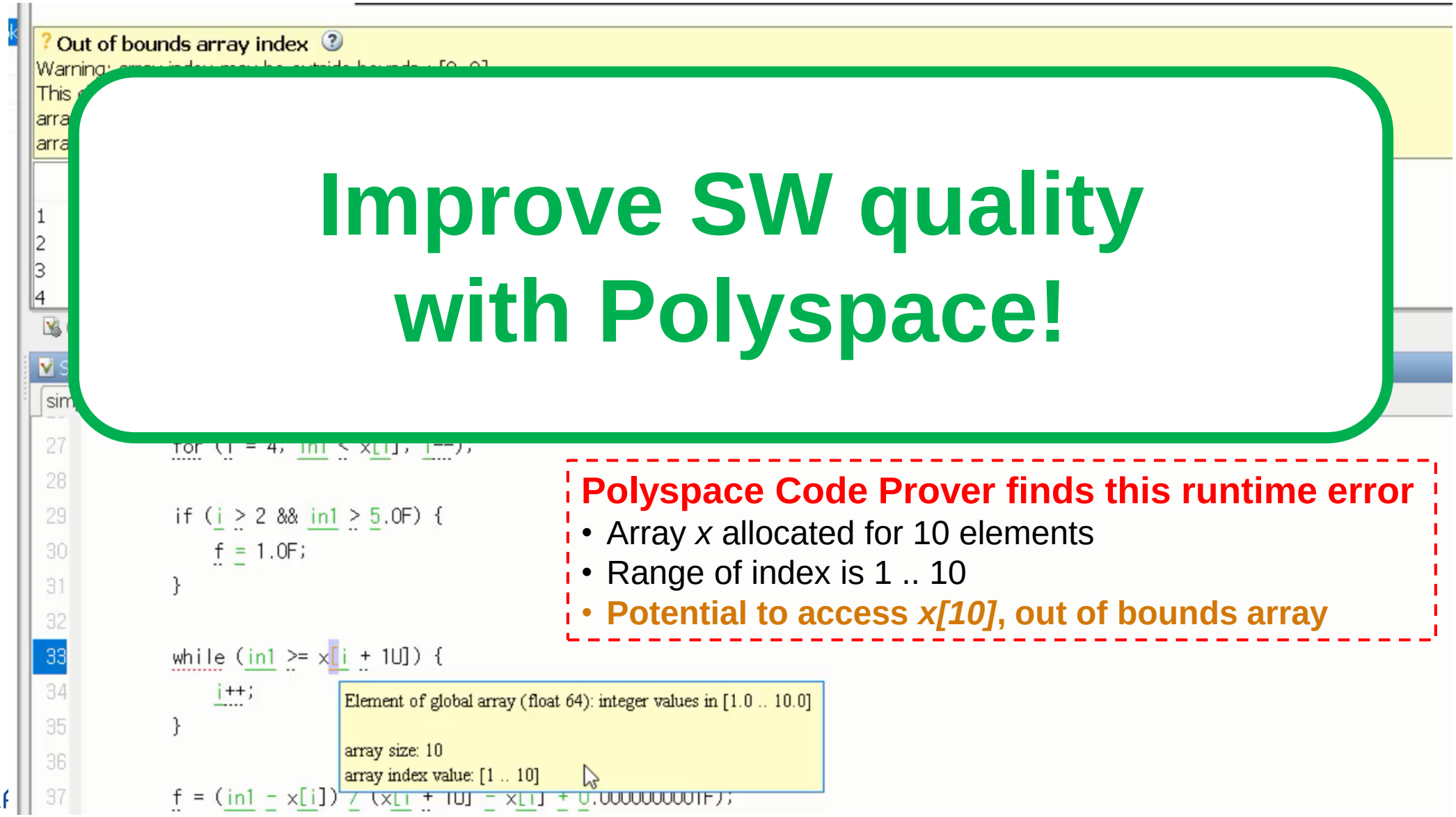
Answer: 6 tests

- 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



Runtime error lurking in the code



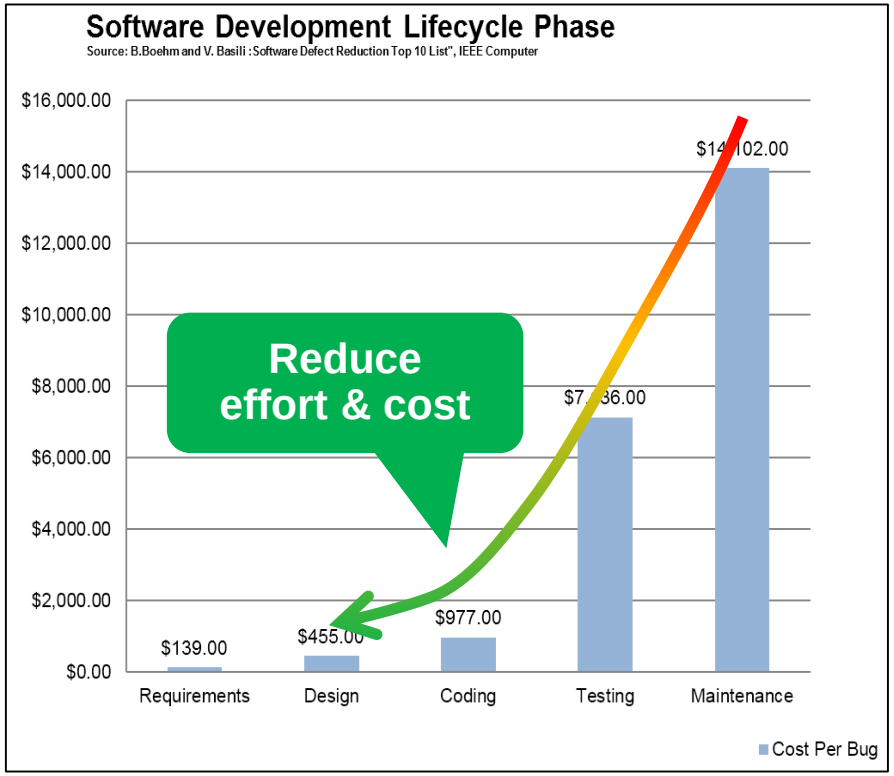
Improve SW quality with Polyspace!

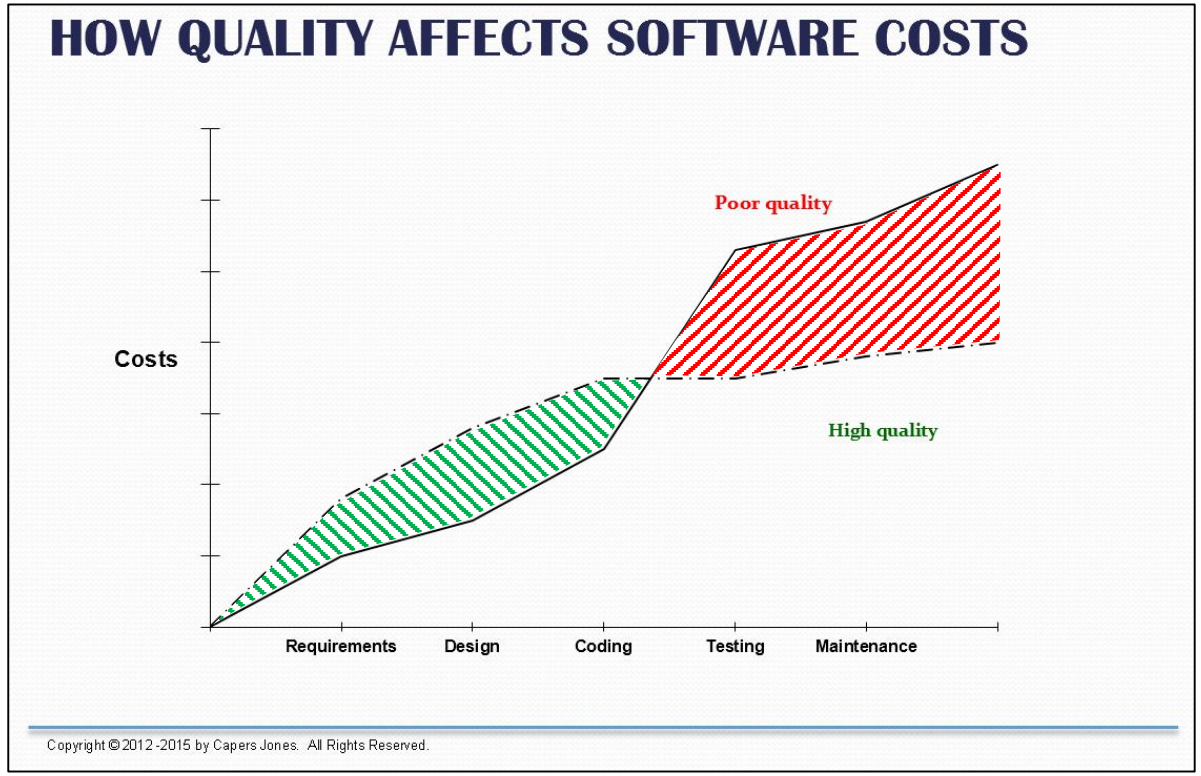
Polyspace Code Prover finds this runtime error

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

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

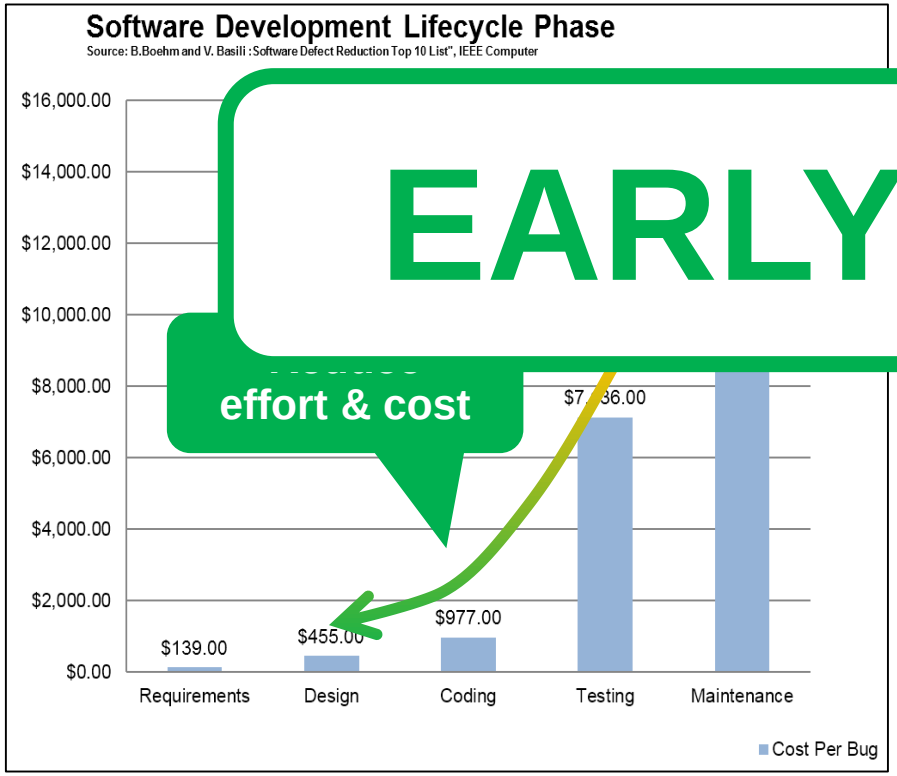


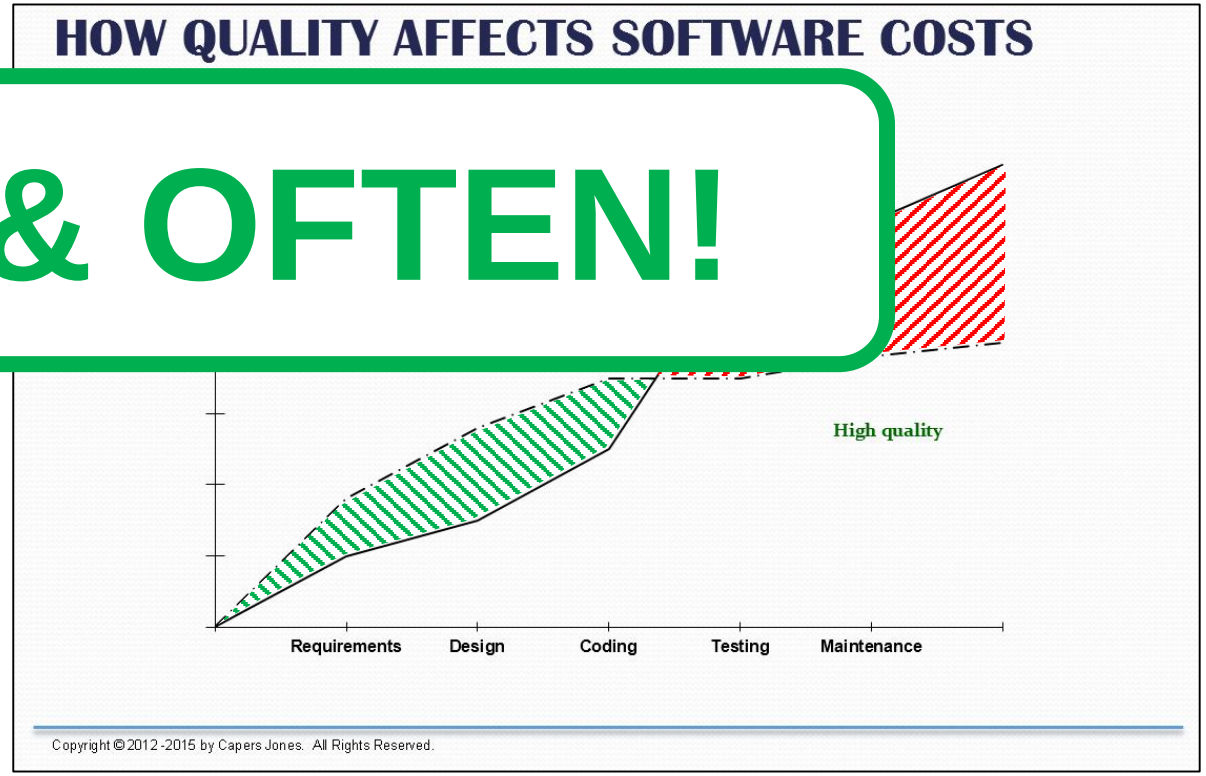


[1] Barry Bohem, "Software Defect Reduction Top 10 List", IEEE
[2] Capers Jones, "Software Quality in 2016: A Survey of the state of the art"

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





EARLY & OFTEN!

[1] Barry Bohem, "Software Defect Reduction Top 10 List", IEEE
[2] Capers Jones, "Software Quality in 2016: A Survey of the state of the art"

Lessons learned from misconceptions

Lessons learned from misconceptions

Polyspace helps you to …

Bug Finder

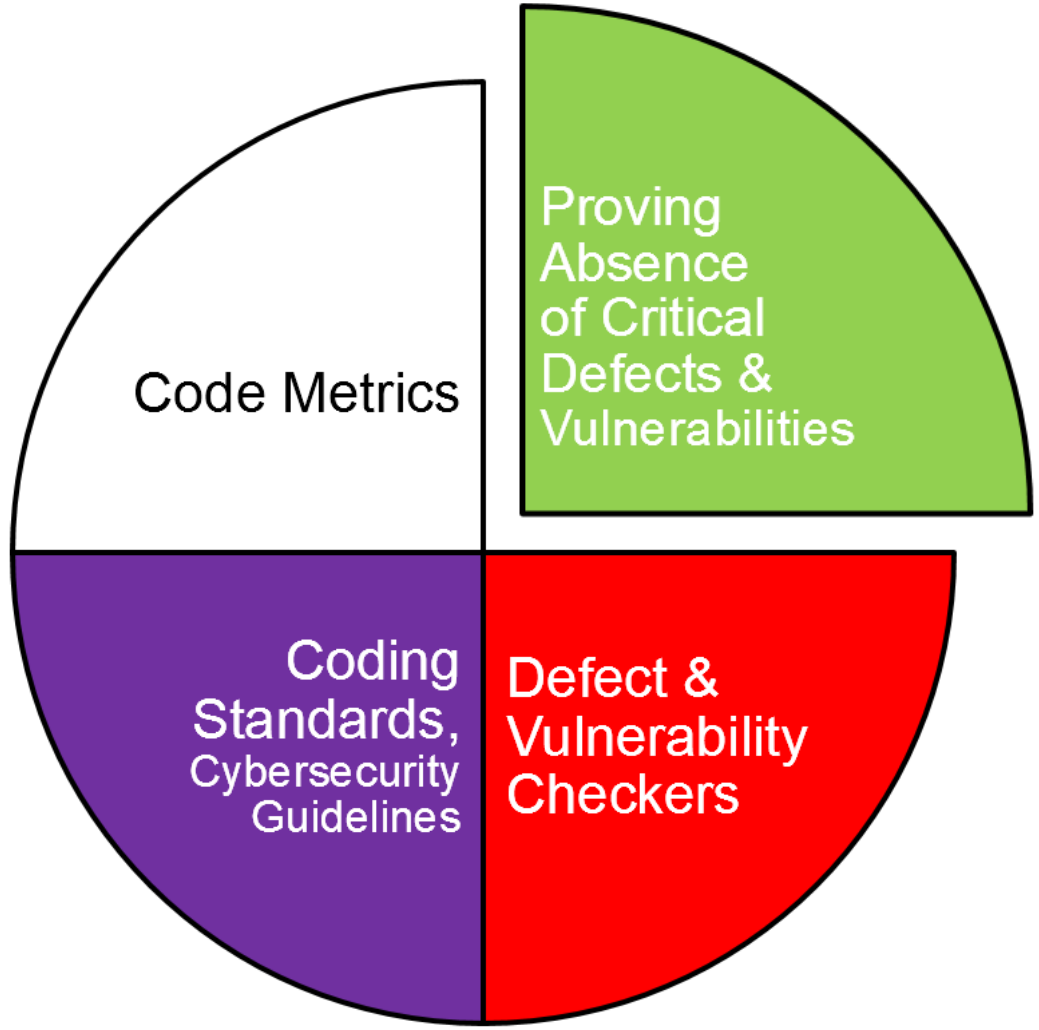
→High Quality, Secure, Compliant Code:

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

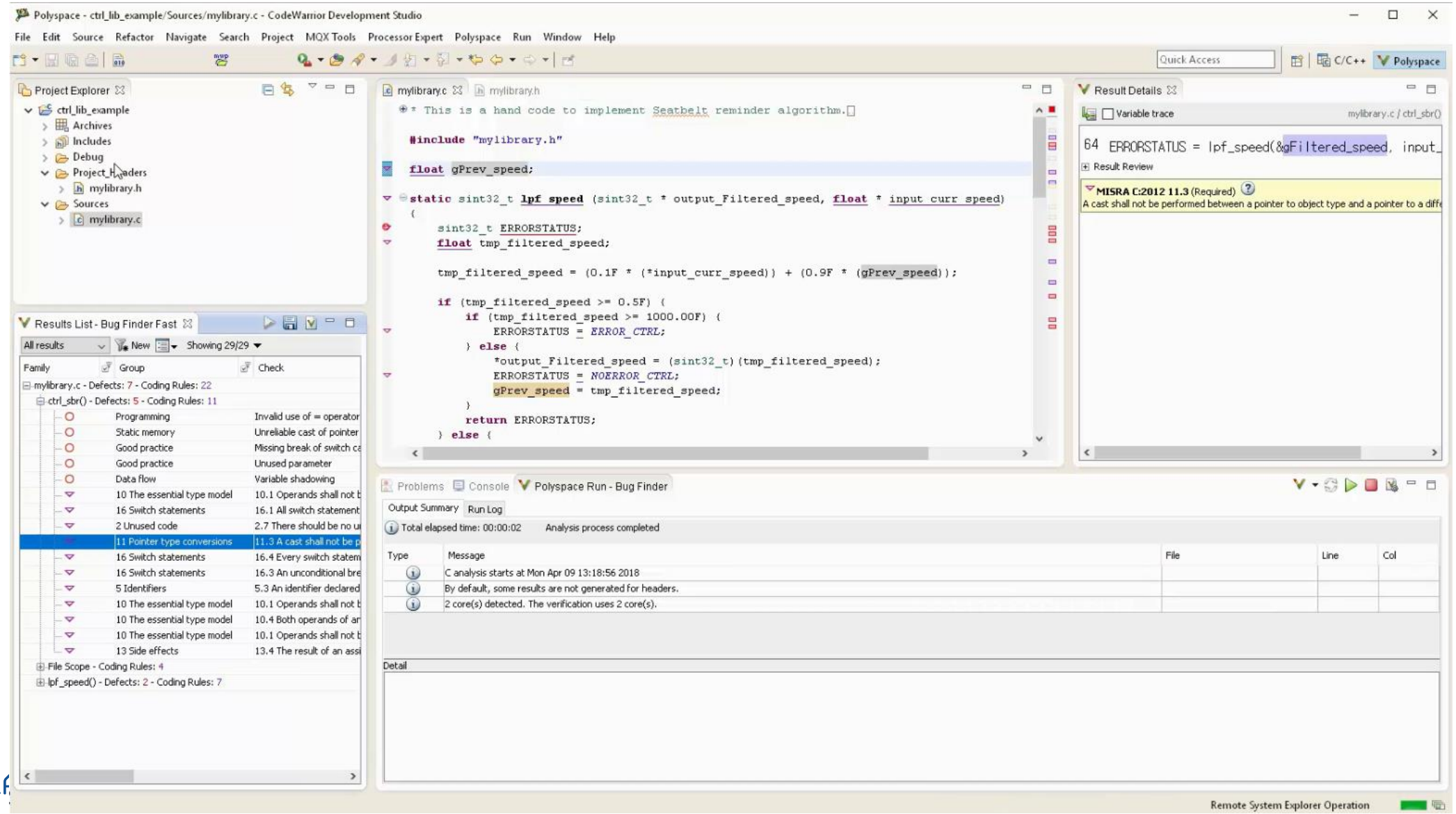
Code Prover

→Fully Trusted Components:

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

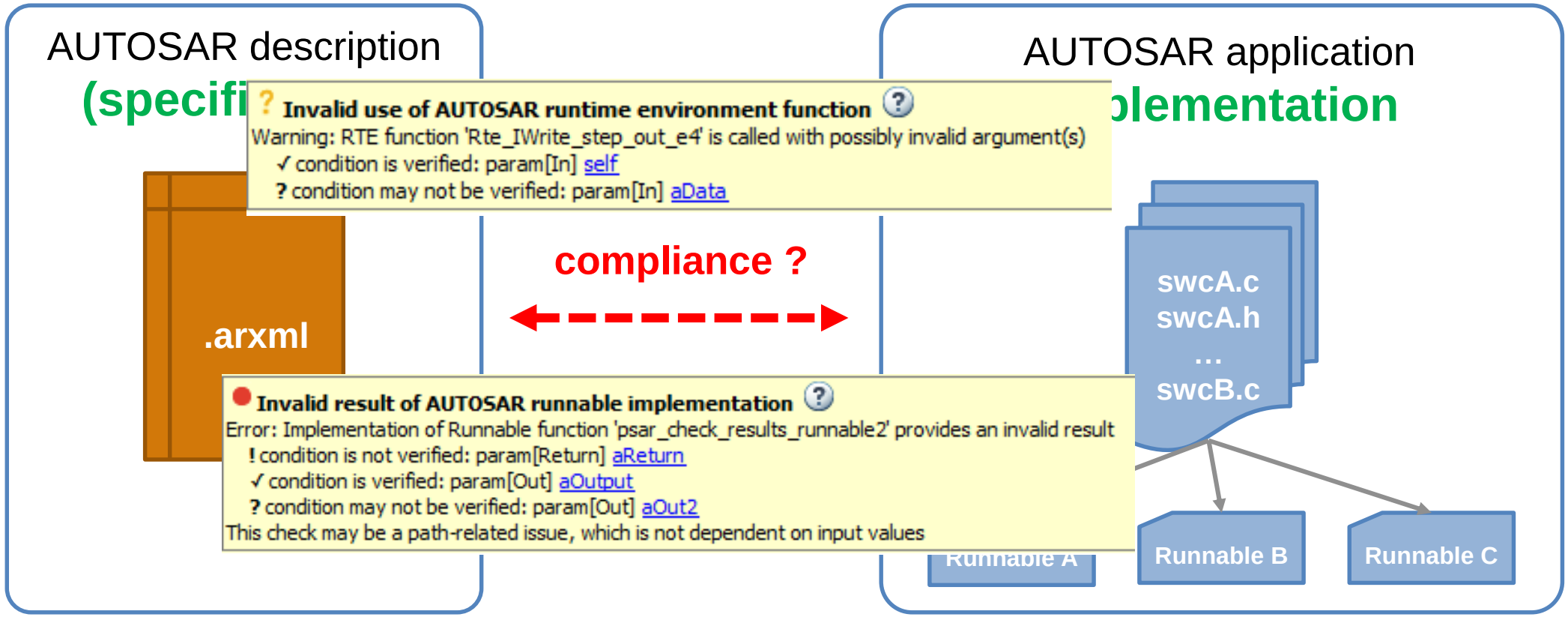


Save-time analysis workflow in Eclipse

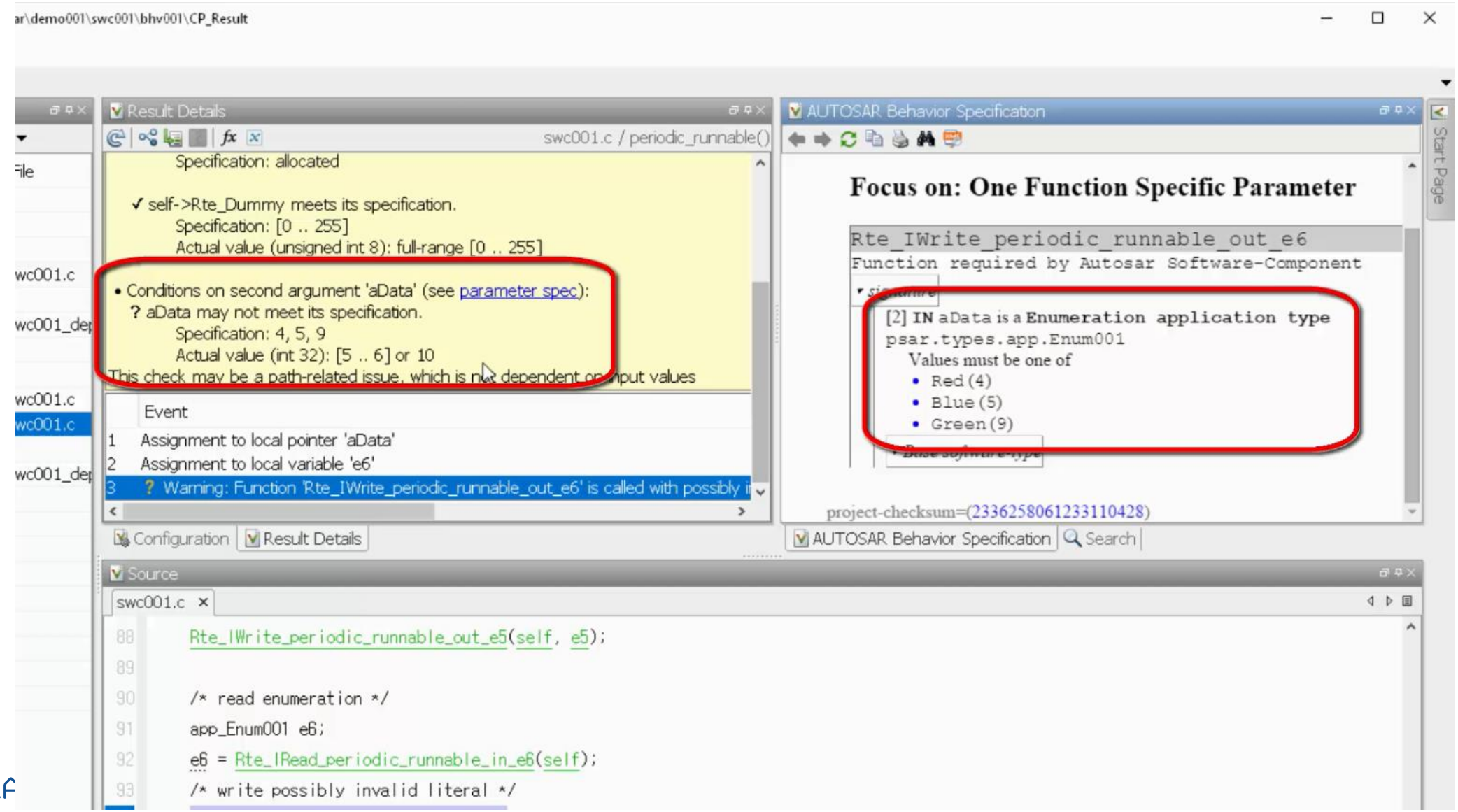


Unit-proving of AUTOSAR components

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

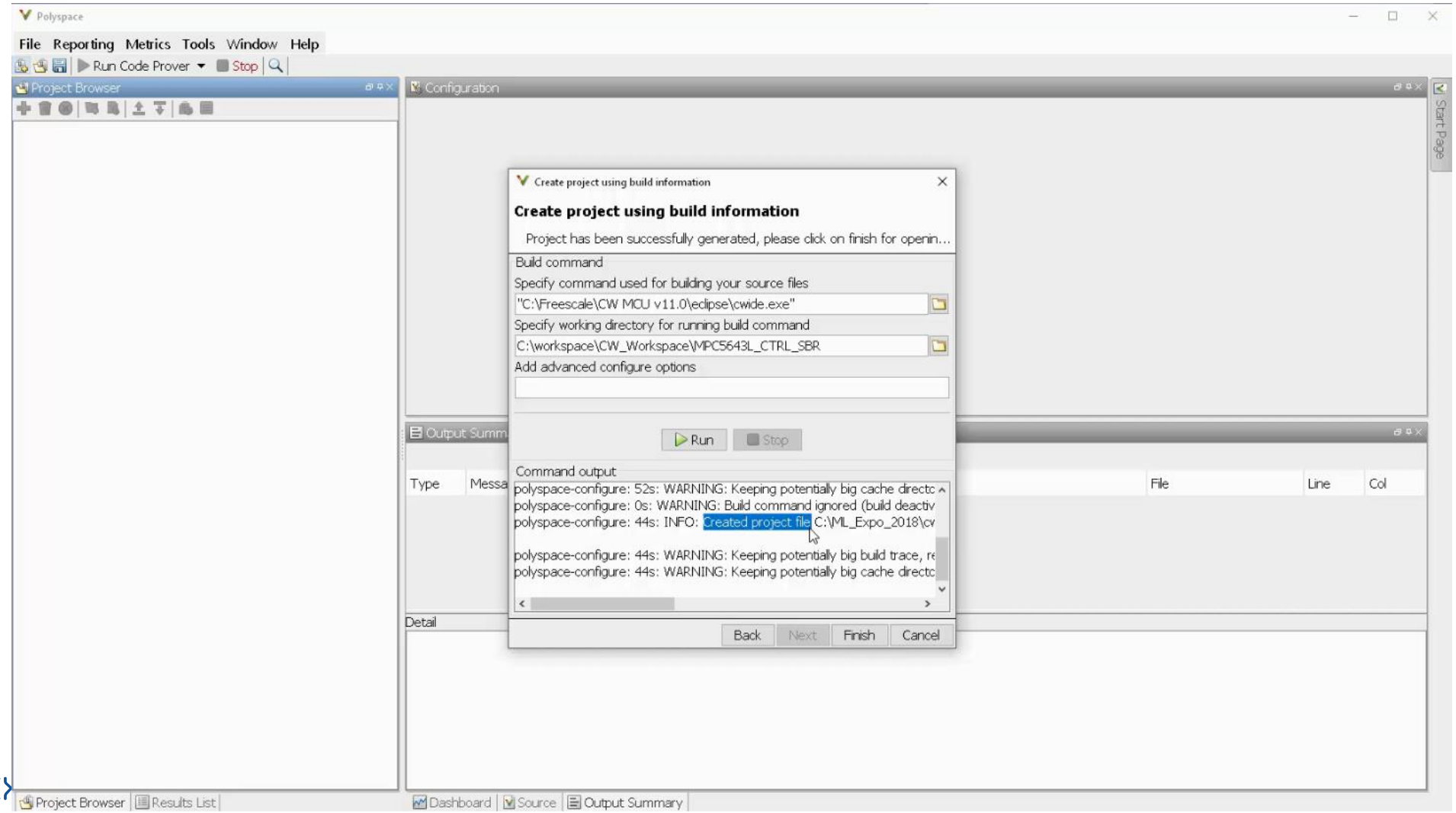


Unit-proving of AUTOSAR components

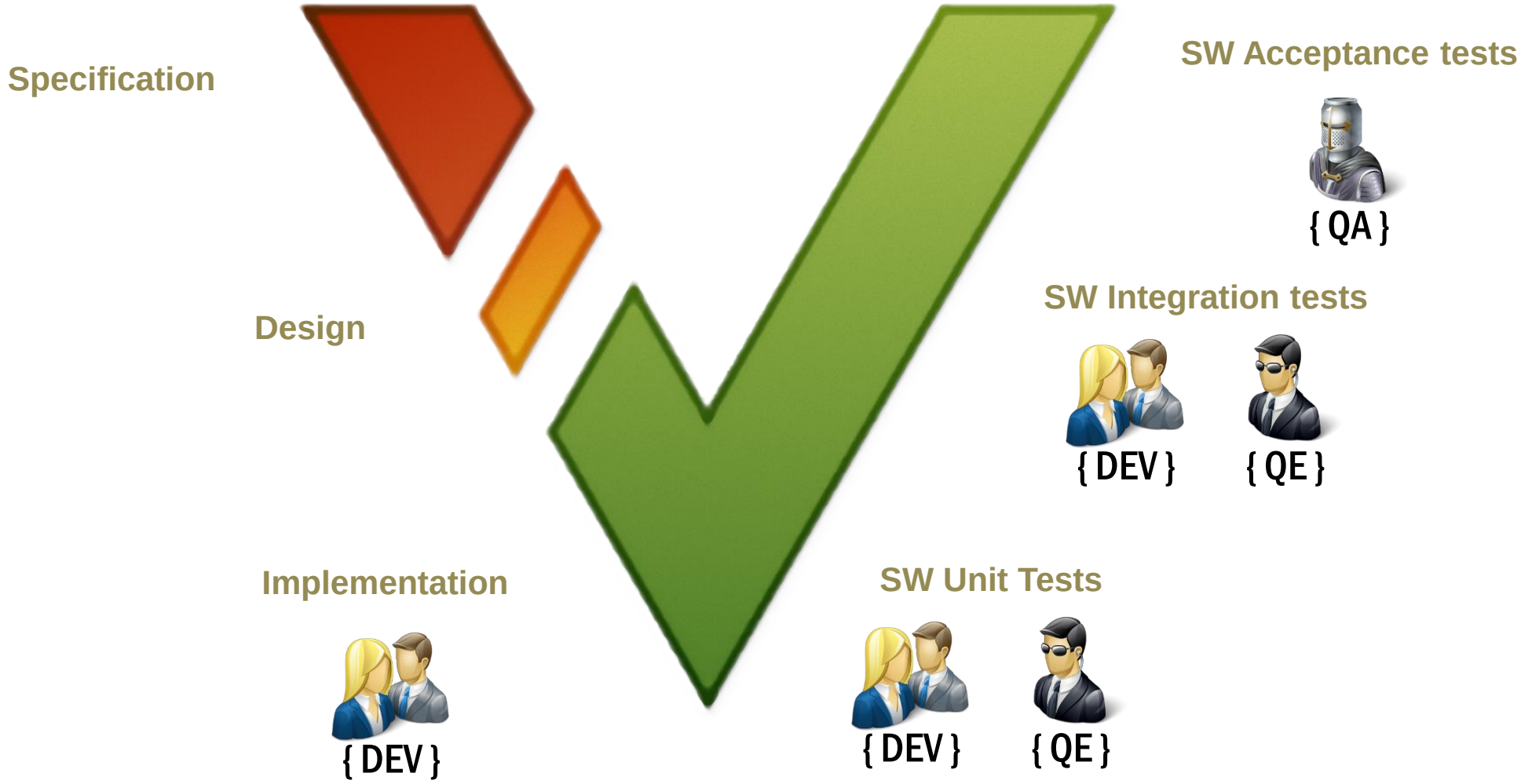


Ease of configuring projects

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



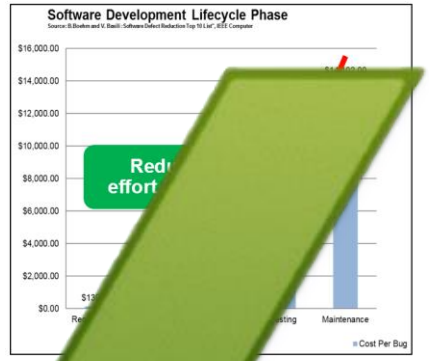
When/Who using Polyspace products?



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