MATLAB EXPO

形式化方法赋能车规级芯片底层软件开发

胡乐华，MathWorks

MathWorks

您将了解到…

- 形式化方法及抽象解释法
- Polyspace对 ISO 26262 和 ISO/SAE 21434 支持
- 形式化方法验证用于性能提升
- 将 Polyspace 应用到开发流程中

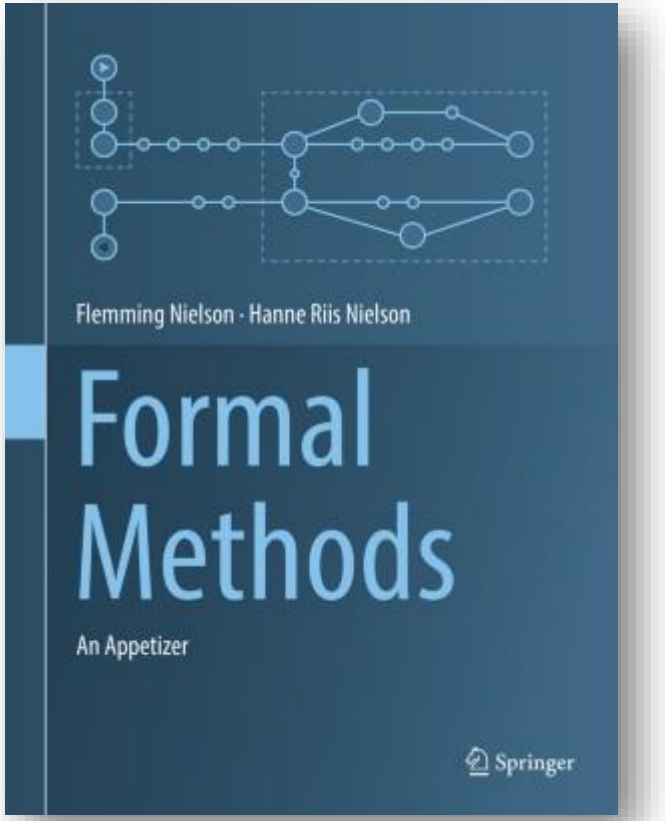
车规级芯片底层软件开发之挑战

- 功能安全和网络安全认证
  - ISO 26262, ISO/SAE 21434
  - CWE, CERT-C
- 芯片运行更多的程序
  - 高效运行驱动程序
  - 权衡防御性编程带来额外的ROM开销
- 更快的交付，响应市场需求
  - 更复杂，开发量更大
  - 协同要求更高

什么是形式化方法和抽象解释法？

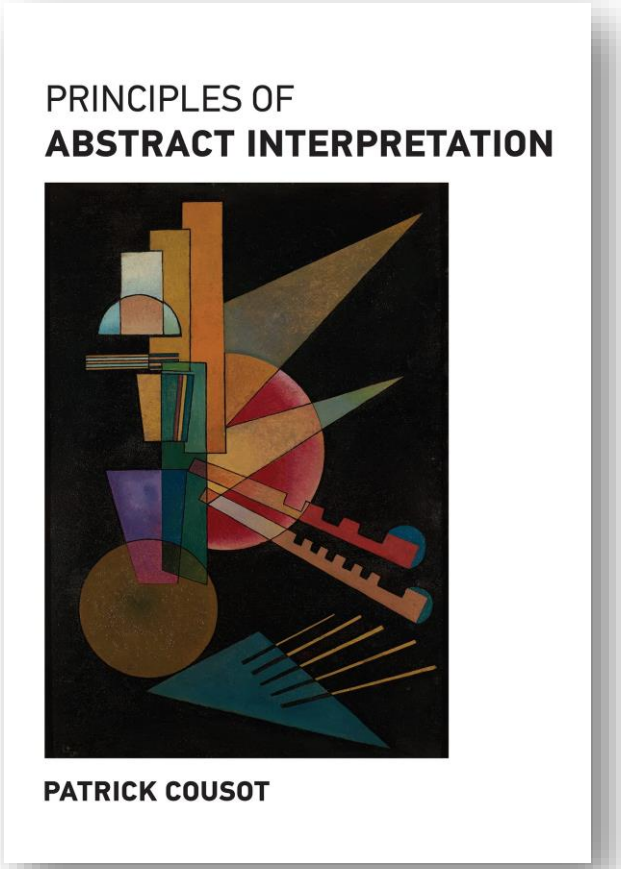
形式化方法

Formal methods are system design techniques that use ***rigorously specified mathematical models*** *to build software and hardware systems. In contrast to other design systems,* ***formal methods use mathematical proof as a complement to system testing in order to ensure correct behavior****. As systems become more complicated, and safety becomes a more important issue, the formal approach to system design offers another level of insurance.*



https://users.ece.cmu.edu/~koopman/des_s99/formal_methods

抽象解释法



In computer science, ***abstract interpretation is a theory of sound approximation of the semantics of computer programs****, based on monotonic functions over ordered sets, especially lattices. It can be viewed as a* ***partial execution of a computer program which gains information about its semantics (e.g., control-flow, data-flow) without performing all the calculations****.*

抽象解释法：证明软件的健壮性

```
1  int new_position(int sensor_pos1, int sensor_pos2)
2  {
3  int actuator_position;
4  int x, y, tmp, magnitude;
5
6  actuator_position = 2; /* default */
7  tmp = 0;               /* values */
8  magnitude = sensor_pos1 / 100;
9  y = magnitude + 5;
10
11 while (actuator_position < 10)
12     {
13     actuator_position++;
14     tmp += sensor_pos2 / 100;
15     y += 3;
16     }
17 if ((3*magnitude + 100) > 43)
18     {
19     magnitude++;
20     x = actuator_position;
21     actuator_position = x / (x - y);
22     }
23 return actuator_position*magnitude + tmp; /* new value */
24 }
```

该行是否有软件bug？
如何证明没有bug？

Polyspace Code Prover 使用抽象解释法证明软件的健壮性

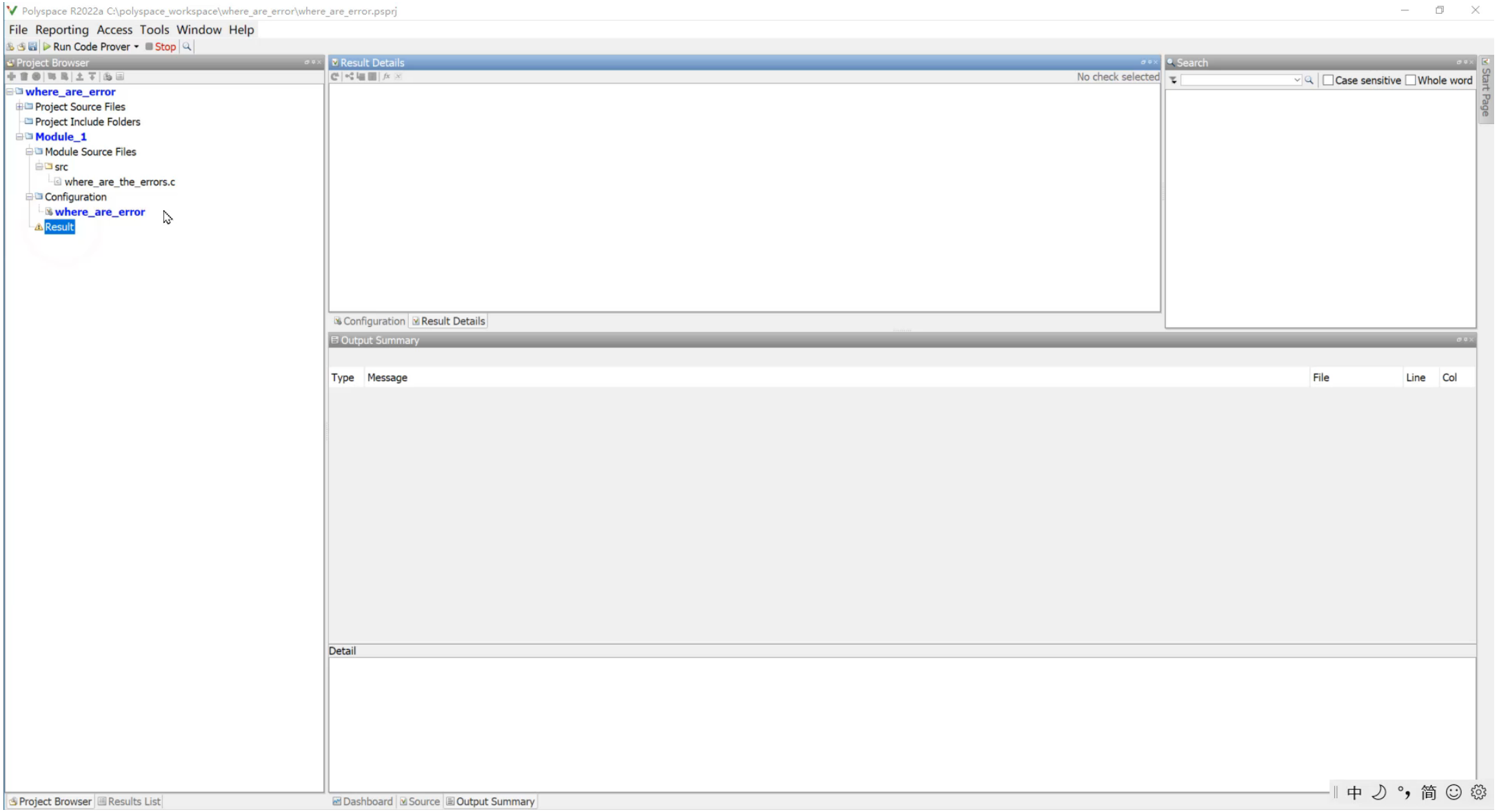
```
int new_position(int sensor_pos1, int sensor_pos2)
{
        int actuator_position;
        int x, y, tmp_pos, magnitude;

        actuator_position = 2; /* default */
        tmp_pos = 0;           /* values */
        magnitude = sensor_pos1 / 100;
        y = magnitude + 5;
        x = actuator_position;

        while (actuator_position < 10)
        {
                actuator_position++;
                tmp_pos += sensor_pos2 / 100;
                y += 3;
        }
        if ((3 * magnitude + 100) > 43)
        {
                magnitude++;
                x = actuator_position;
                actuator_position = x / (x - y);
        }
        return actuator_position + tmp_pos;
}
```

```
operator - on type int 32
    left:  10
    right: [11 .. 21474865 (0x147AE31)]
    result: [-21474855 .. -1]
    (result is truncated)
                                Press 'F2' for focus
```

证明除法安全

除数永不为零

示例：Polyspace Code Prover



Polyspace 代码着色让问题暴露无遗

any

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

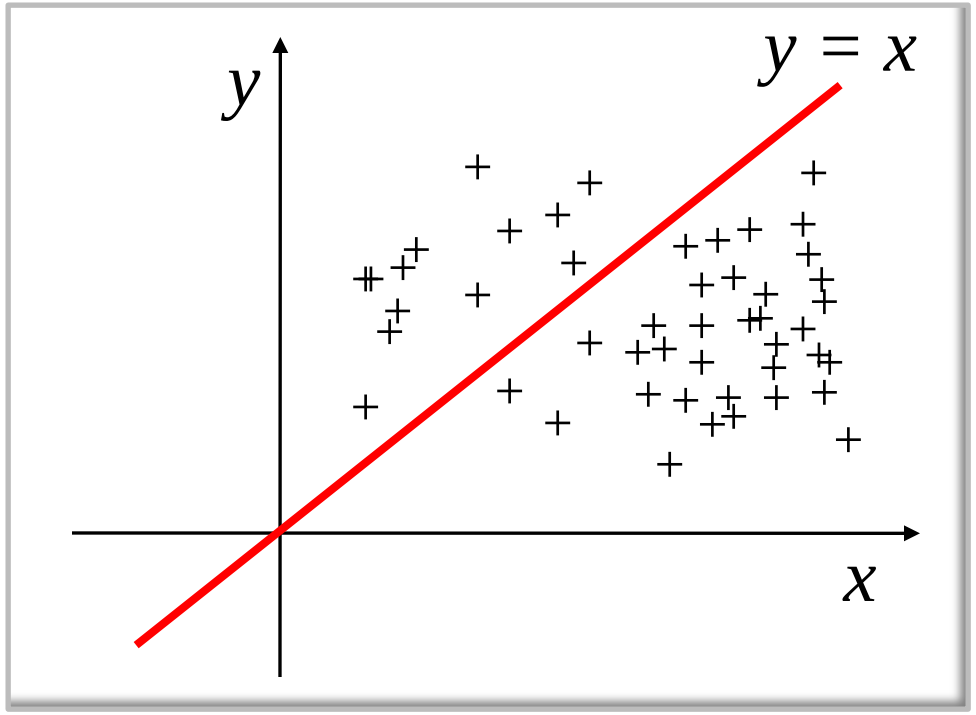
Range data
tool tip

```
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]

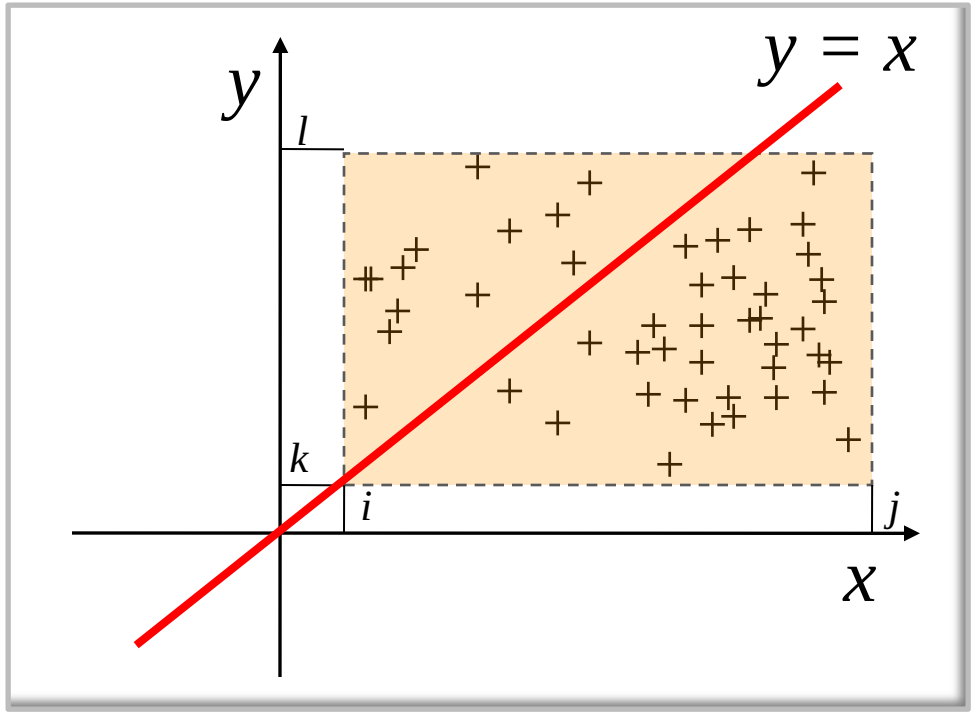
理解抽象解释法

x 和 y 可以是任何值



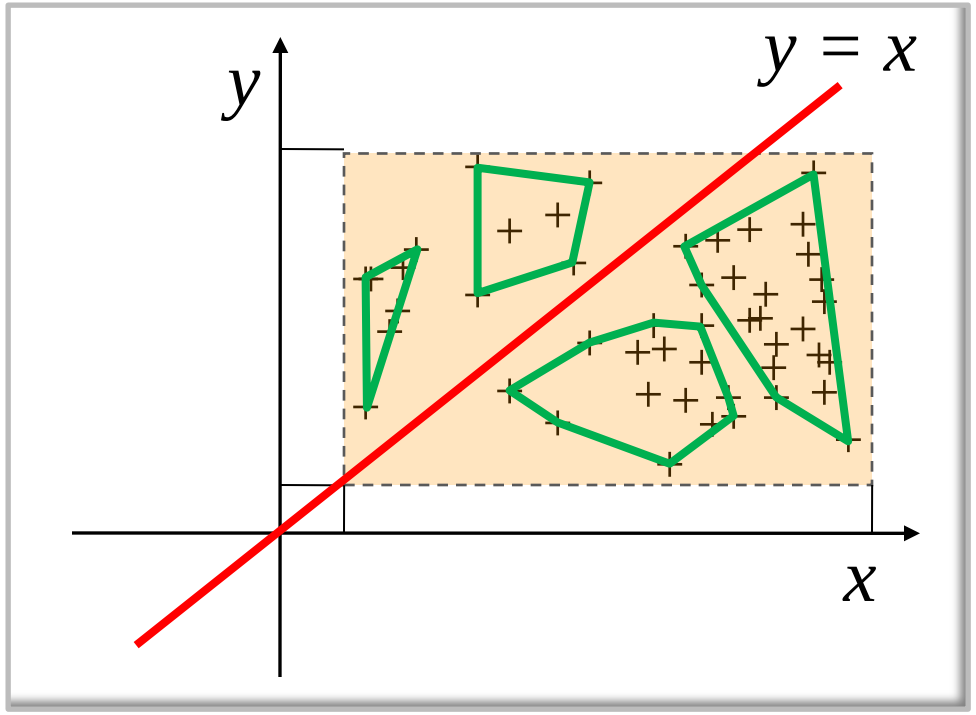
理解抽象解释法

类型分析，界定范围



理解抽象解释法

抽象解释法



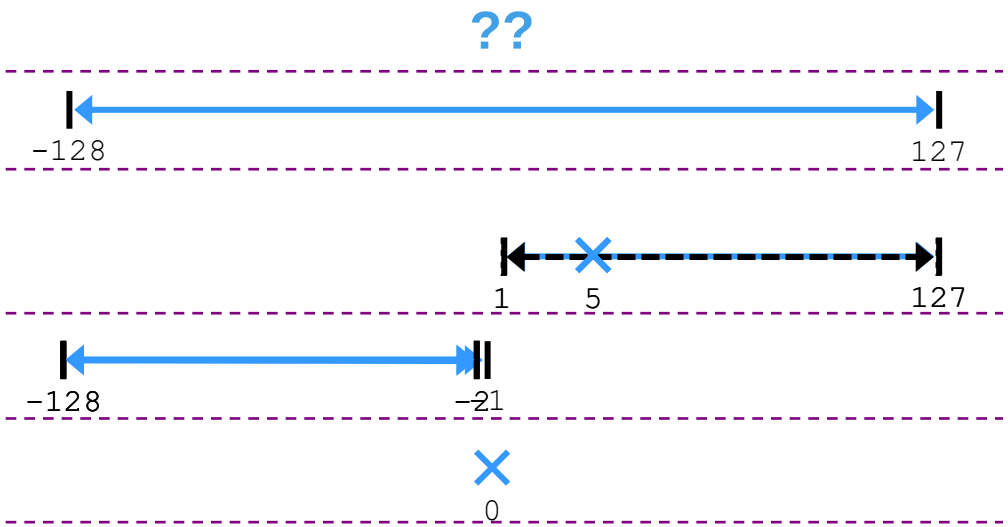
不运行

无需测试用例

理解抽象解释法

抽象解释法考虑所有可能输入，所有可能执行路径来证明是否有缺陷。

```
signed char x, y;

x = random_char();

if (x > 0) {
    x = 5;
} else if (x != 0){
    y = 100 / (x + 1);
} else {
    y = 200 / x;
}

printf("%d",x);
```



完整的静态分析能力

Bug Finder

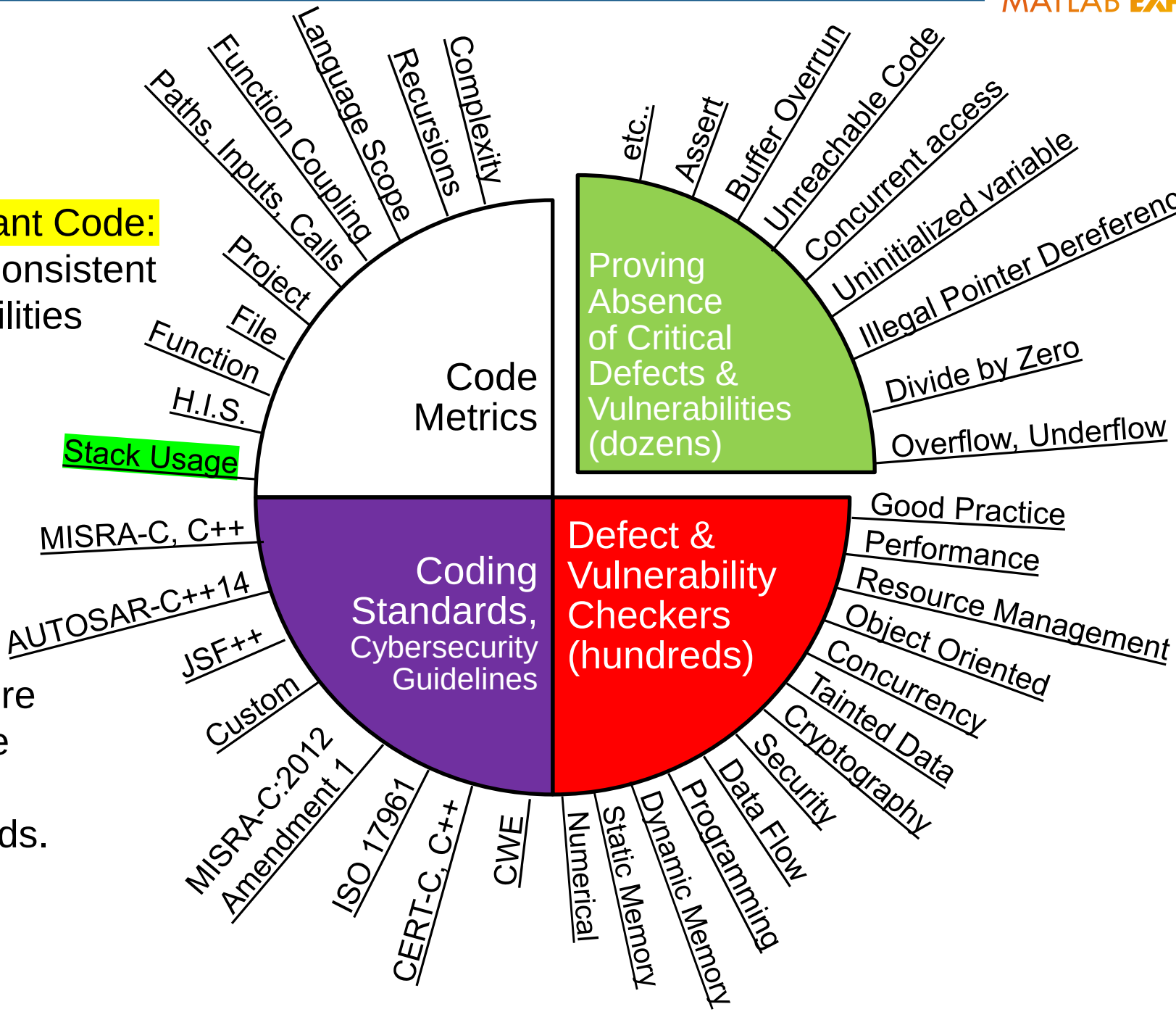
→High Quality, Secure, Compliant Code:

- Measurable, Maintainable, Consistent
- Very few defects or vulnerabilities
- Credits for functional safety, cybersecurity standards.

Code Prover

→Fully Trusted Components:

- Reliable, Robust, Safe, Secure
- Proven free of critical runtime defects and vulnerabilities
- Additional credits for standards.



Polyspace 支持功能安全 ISO 26262 最高等级认证

Polyspace Bug Finder 对ISO 26262 Part6要求的覆盖

Table 1 - Topics To Be Covered By Modeling and Coding Guidelines

	Topics	ASIL A	B	C	D
1i	Concurrency aspect	+	+	+	+
1h	Use of naming conventions	‡	‡	‡	‡
1g	Use of style guides	+	‡	‡	‡
1f	Use of unambiguous graphical representation	+	‡	‡	‡
1e	Use of well-trusted design principles	+	+	‡	‡
1d	Use of defensive implementation techniques	+	+	‡	‡
1c	Enforcement of strong typing	‡	‡	‡	‡
1b	Use of language subsets	‡	‡	‡	‡
1a	Enforcement of low complexity	‡	‡	‡	‡

Table 3 – Principles for Software Architectural Design

	Topics	ASIL A	B	C	D
1i	Appropriate management of shared resources	‡	‡	‡	‡
(...)					
1e	Loose coupling between software components	+	‡	‡	‡
(...)					
1c	Restricted size of interfaces	+	+	+	‡
1b	Restricted size and complexity of software components	‡	‡	‡	‡
(...)					

Table 6 – Design Principles for Software Unit Design and Implementation

	Topics	ASIL A	B	C	D
1j	No recursions	+	+	‡	‡
1i	No unconditional jumps	‡	‡	‡	‡
1h	No hidden data flow or control flow	+	‡	‡	‡
1g	No implicit type conversions	+	‡	‡	‡
1f	Restricted use of pointers	+	‡	‡	‡
1e	Avoid global variables or else justify their usage	+	+	‡	‡
1d	No multiple use of variable names	‡	‡	‡	‡
1c	Initialization of variables	‡	‡	‡	‡
1b	No dynamic objects or variables, or else online test during their creation	+	‡	‡	‡
1a	One entry and one exit point in subprograms and functions	‡	‡	‡	‡

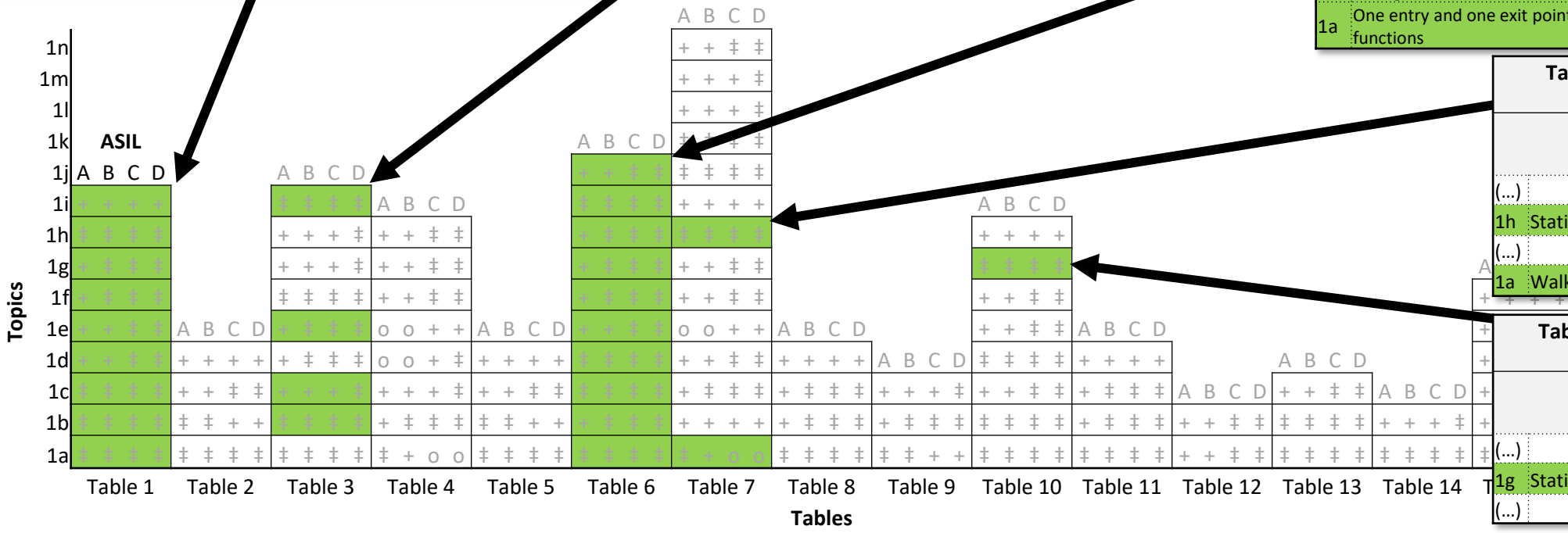


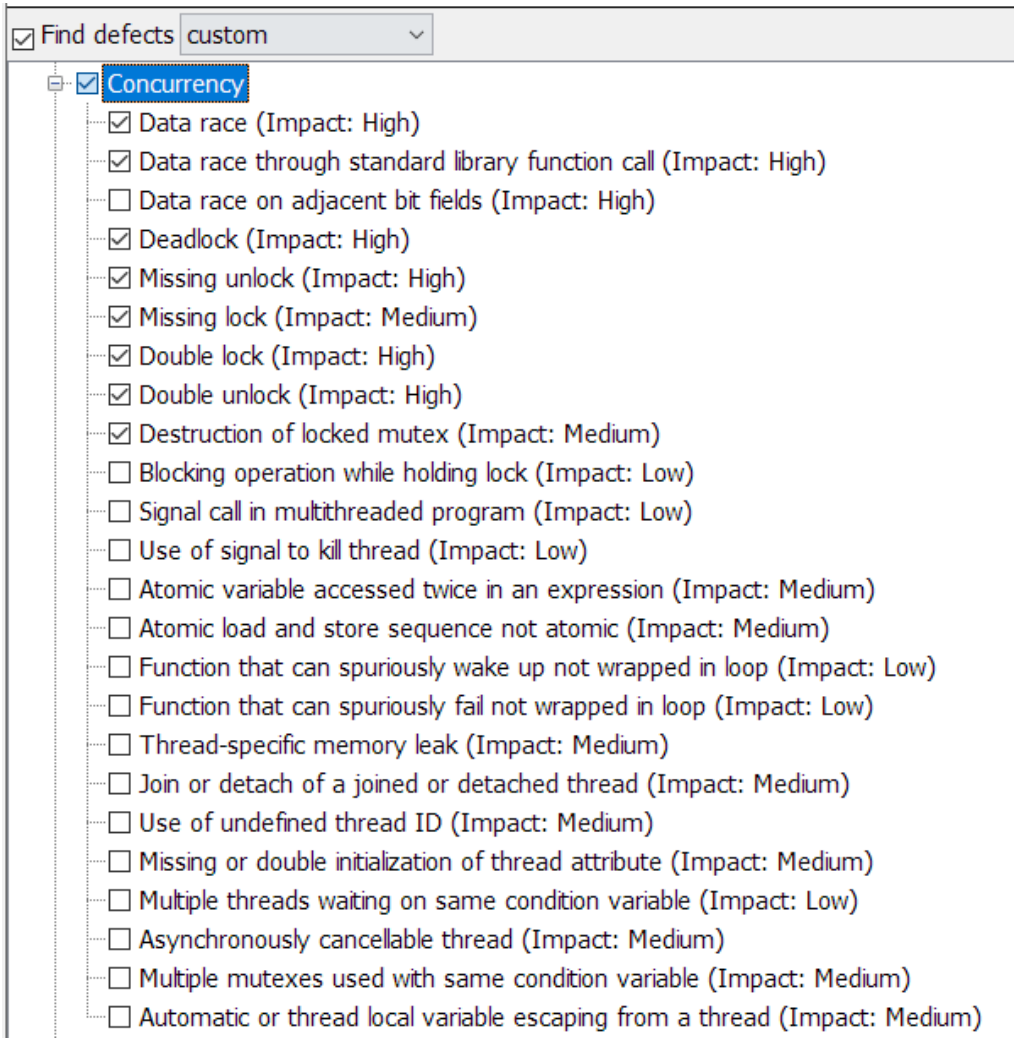
Table 7 – Methods for Software Unit Verification

	Topics	ASIL A	B	C	D
(...)					
1h	Static code analysis	‡	‡	‡	‡
(...)					
1a	Walk-through	‡	+	o	o

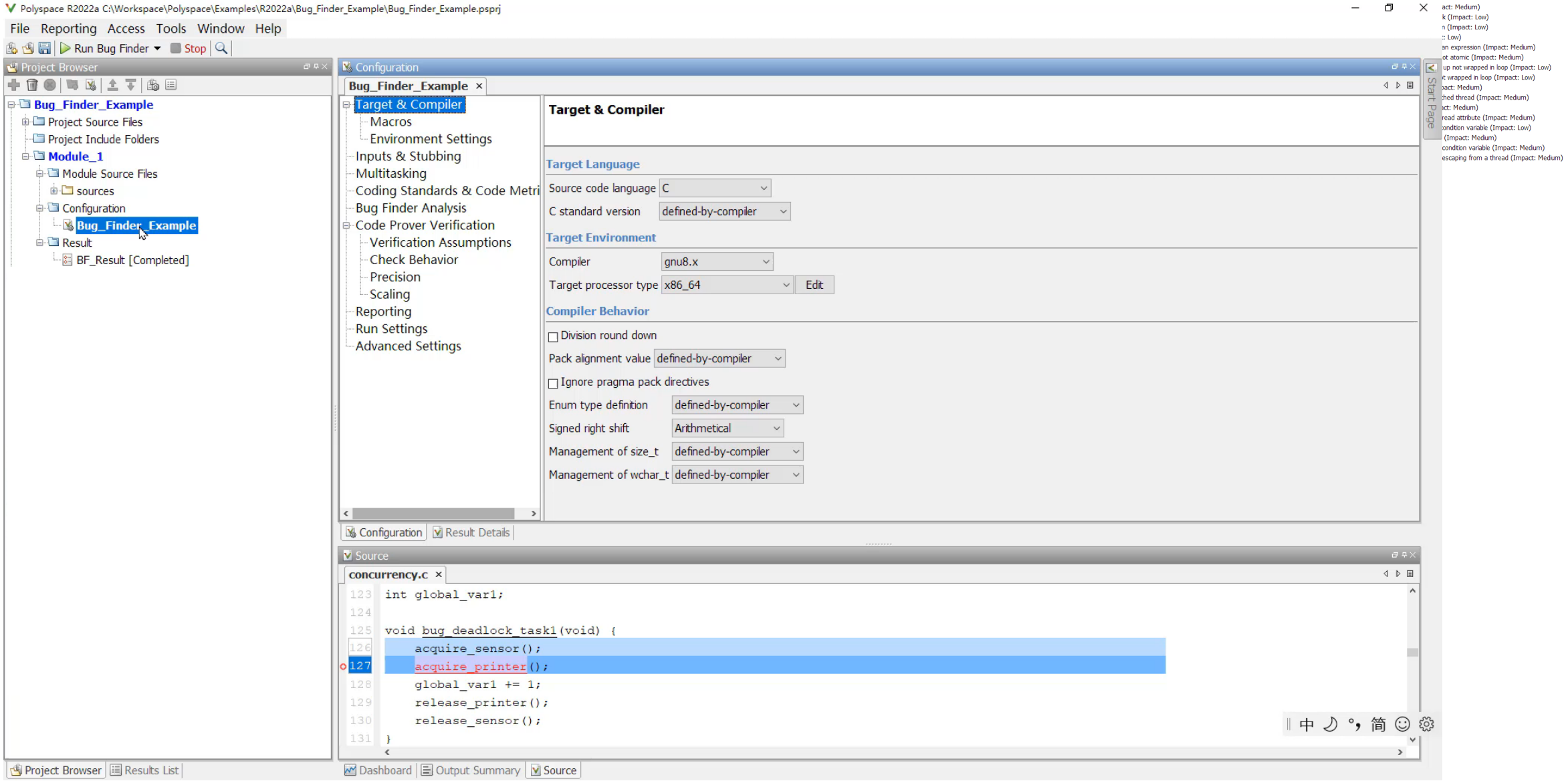
Table 10 – Methods for Verification of Software Integration

	Topics	ASIL A	B	C	D
(...)					
1g	Static code analysis	‡	‡	‡	‡
(...)					

Polyspace Bug Finder 解决并发问题



Polyspace Bug Finder 死锁问题检查



Polyspace Code Prover ISO 26262 Part 6进一步覆盖要求和增强…

Table 1 - Topics To Be Covered By Modeling and Coding Guidelines

	Topics	ASIL A	B	C	D
1i	Concurency aspects (more…)	+	+	+	+
(…)					

Table 4 – Methods for the Verification of the Software Architectural Design

	Topics	ASIL A	B	C	D
(…)					
1g	Data flow analysis	+	+	‡	‡
1f	Control flow analysis	+	+	‡	‡
1e	Formal verification	o	o	+	+
(…)					

Table 6 – Design Principles for Software Unit Design and Implementation

	Topics	ASIL A	B	C	D
1j	No recursions (more…)	+	+	‡	‡
(…)					
1g	No implicit type conversions (more…)	+	‡	‡	‡
1f	Restricted use of pointers (more…)	+	‡	‡	‡
1e	Avoid global variables or else justify their usage (more…)	+	+	‡	‡
(…)					
1c	Initialization of variables (more…)	‡	‡	‡	‡
(…)					

Table 7 – Methods for Software Unit Verification

	Topics	ASIL A	B	C	D
(…)					
1g	Data flow analysis	+	+	‡	‡
1f	Control flow analysis	+	+	‡	‡
1e	Formal verification	o	o	+	+
(…)					
1a	Walk-through (more…)	‡	+	o	o

Table 3 – Principles for Software Architectural Design

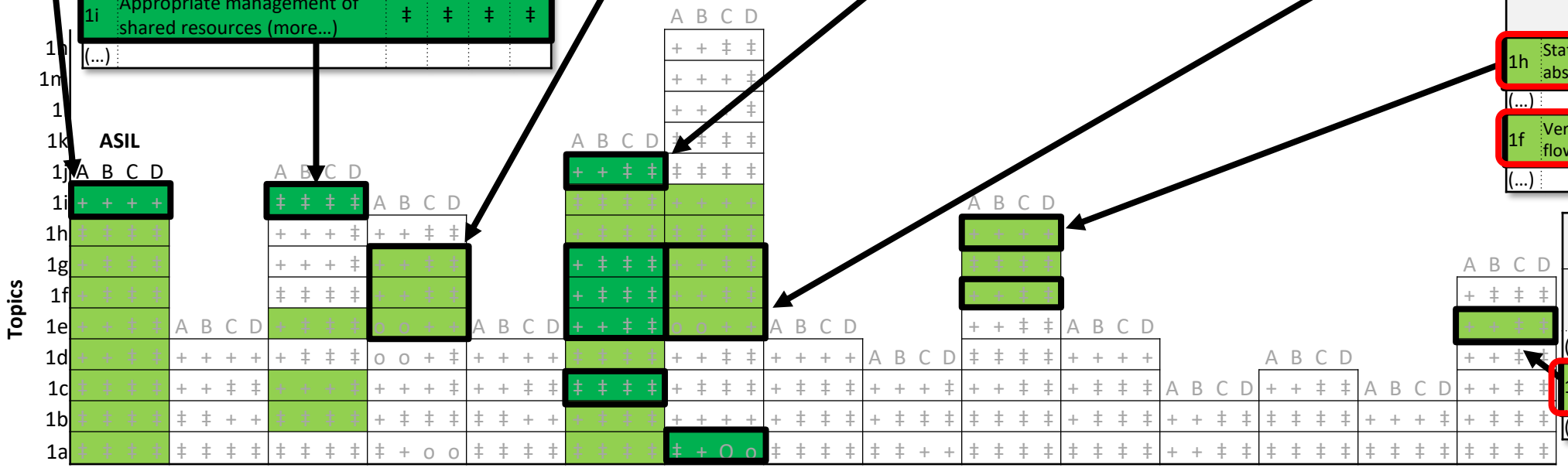
	Topics	ASIL A	B	C	D
1i	Appropriate management of shared resources (more…)	‡	‡	‡	‡
(…)					

Table 10 – Methods for Verification of Software Integration

	Topics	ASIL A	B	C	D
1h	Static analysis based on abstract interpretation	+	+	+	+
(…)					
1f	Verification of the control flow and data flow	+	+	‡	‡
(…)					

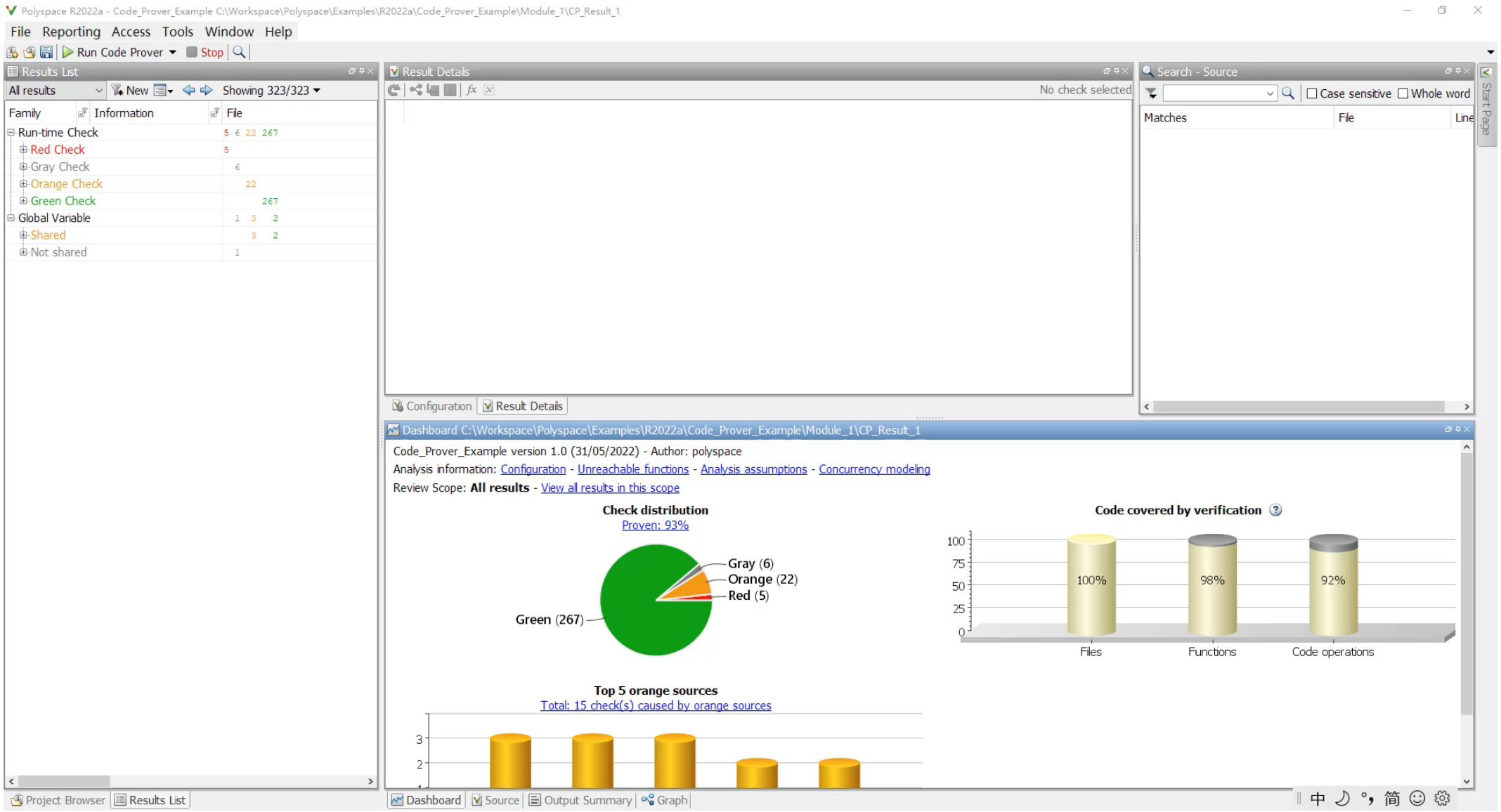
Table 15 – Methods for Deriving Test Cases for Software Integration Testing

	Topics	ASIL A	B	C	D
(…)					
1e	Analysis of functional dependencies	+	+	‡	‡
(…)					

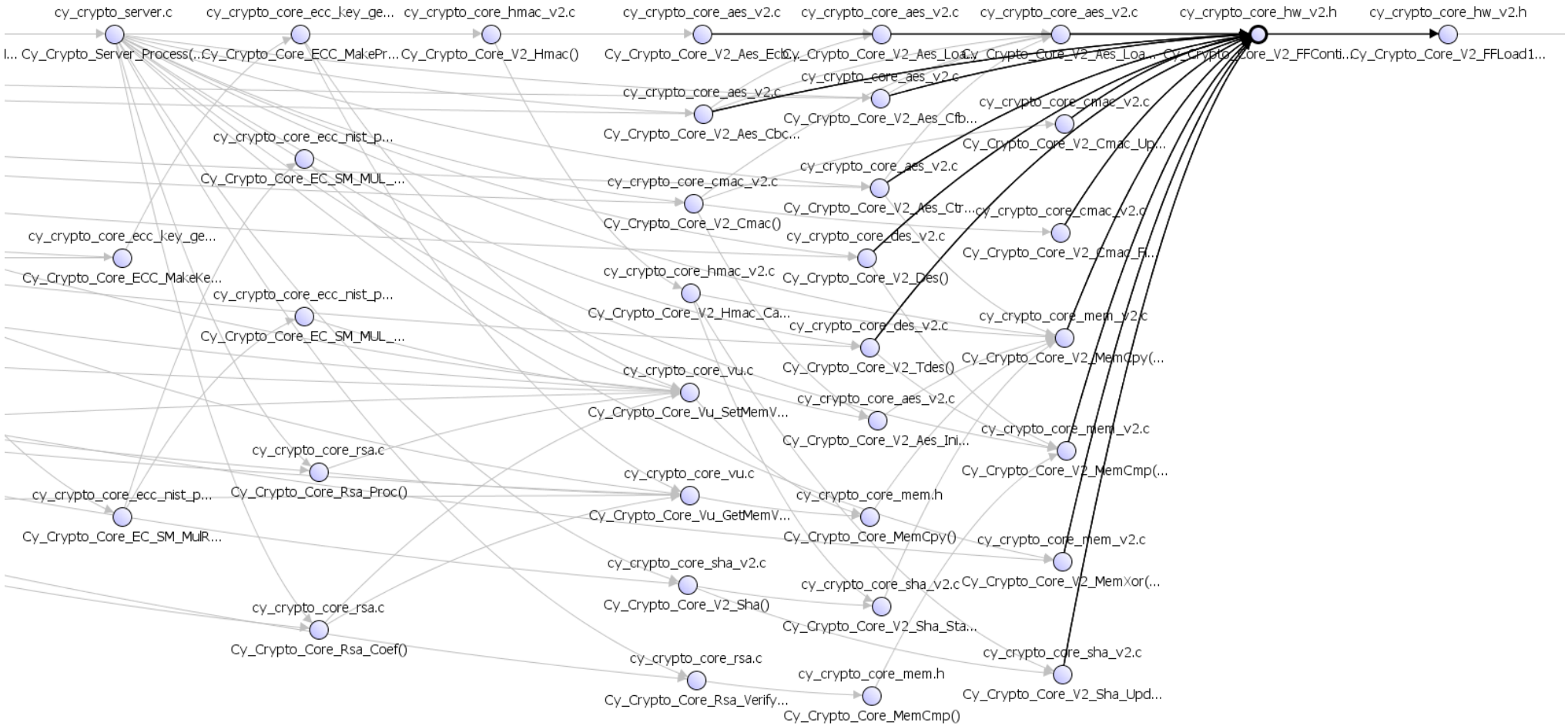


Key:
- Bug Finder Credit
- Code Prover Credit
- Code Prover Enhances Bug Finder

Polyspace Code Prover 证明变量访问冲突

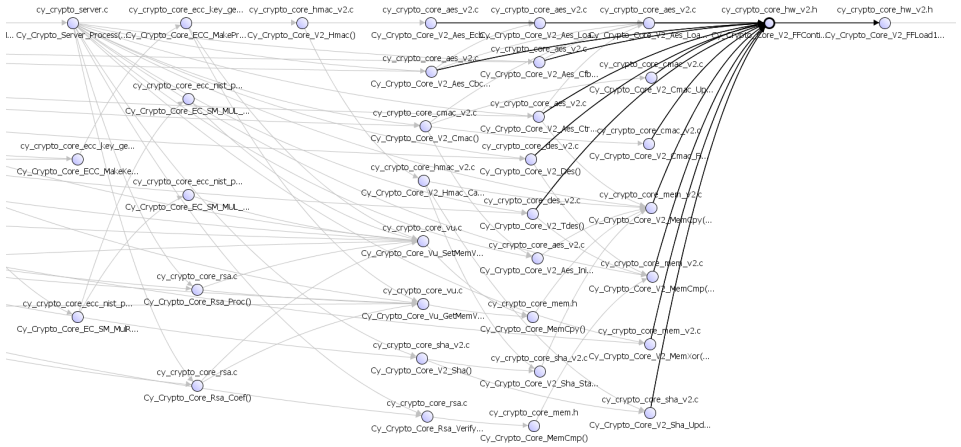


从 Code Prover 提供控制流和数据流分析



问题追溯树

从 Code Prover 提供控制流和数据流分析

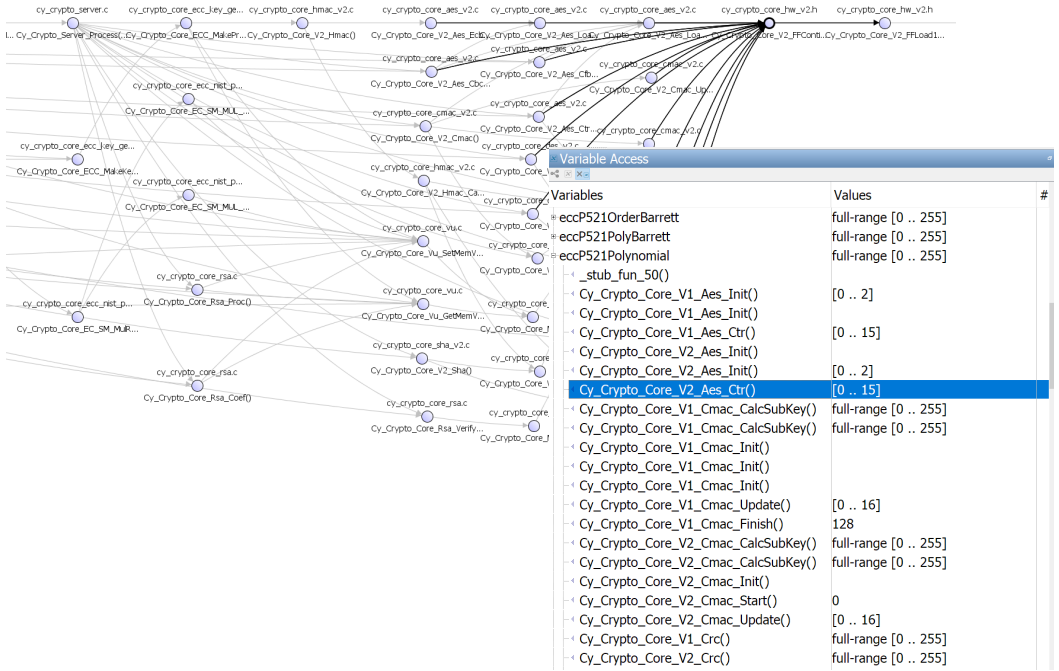


Variable Access

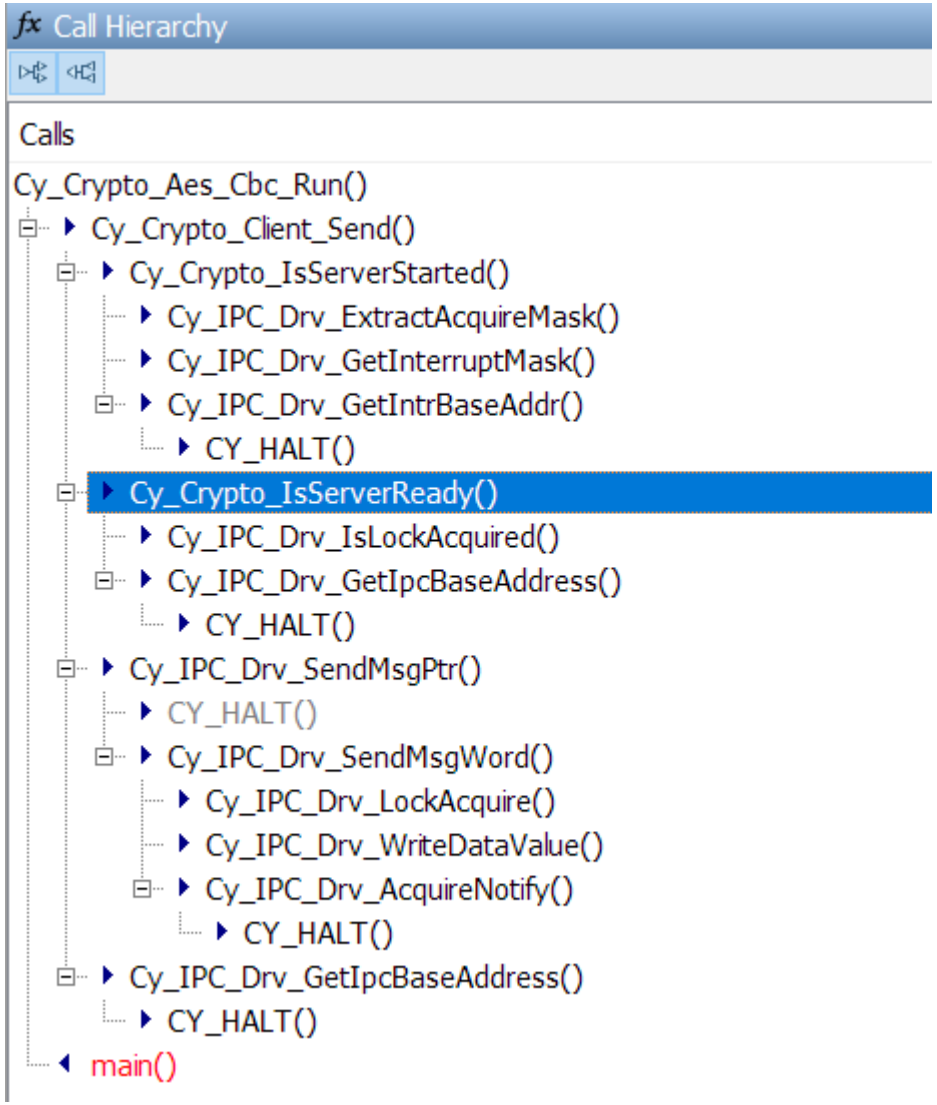
Variables	Values	#
eccP521OrderBarrett	full-range [0 .. 255]	
eccP521PolyBarrett	full-range [0 .. 255]	
eccP521Polynomial	full-range [0 .. 255]	
_stub_fun_50()		
Crypto_Core_V1_Aes_Init()	[0 .. 2]	
Crypto_Core_V1_Aes_Init()		
Crypto_Core_V1_Aes_Ctr()	[0 .. 15]	
Crypto_Core_V2_Aes_Init()		
Crypto_Core_V2_Aes_Init()	[0 .. 2]	
Crypto_Core_V2_Aes_Ctr()	[0 .. 15]	
Crypto_Core_V1_Cmac_CalcSubKey()	full-range [0 .. 255]	
Crypto_Core_V1_Cmac_CalcSubKey()	full-range [0 .. 255]	
Crypto_Core_V1_Cmac_Init()		
Crypto_Core_V1_Cmac_Init()		
Crypto_Core_V1_Cmac_Init()		
Crypto_Core_V1_Cmac_Update()	[0 .. 16]	
Crypto_Core_V1_Cmac_Finish()	128	
Crypto_Core_V2_Cmac_CalcSubKey()	full-range [0 .. 255]	
Crypto_Core_V2_Cmac_CalcSubKey()	full-range [0 .. 255]	
Crypto_Core_V2_Cmac_Init()		
Crypto_Core_V2_Cmac_Start()	0	
Crypto_Core_V2_Cmac_Update()	[0 .. 16]	
Crypto_Core_V1_Crc()	full-range [0 .. 255]	
Crypto_Core_V2_Crc()	full-range [0 .. 255]	

变量访问树

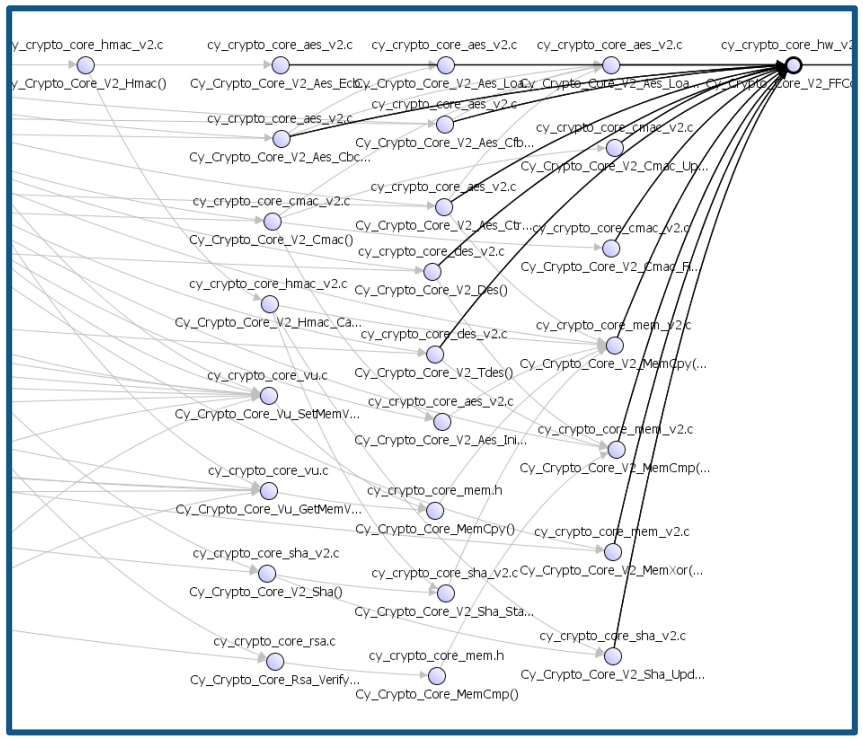
从 Code Prover 提供控制流和数据流分析

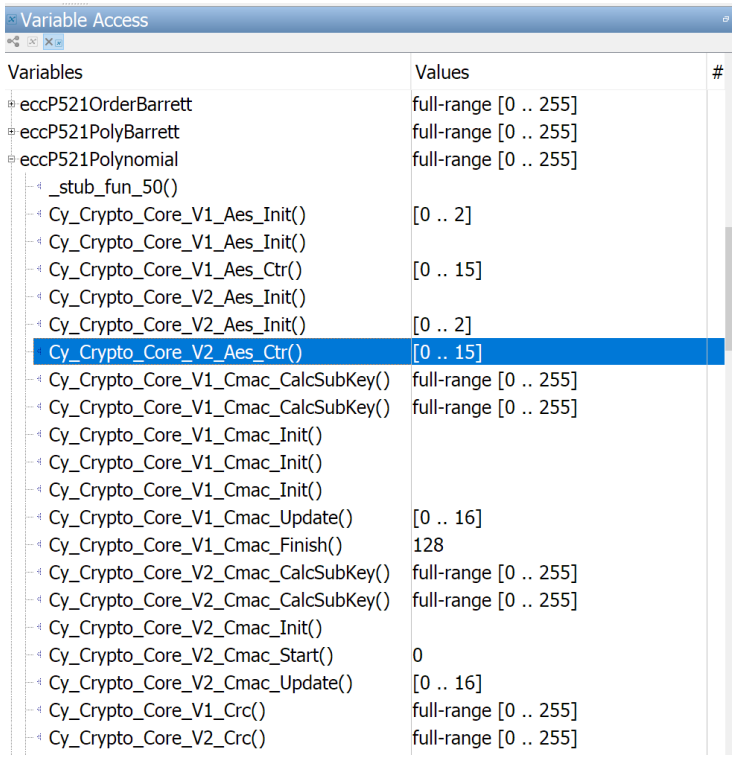


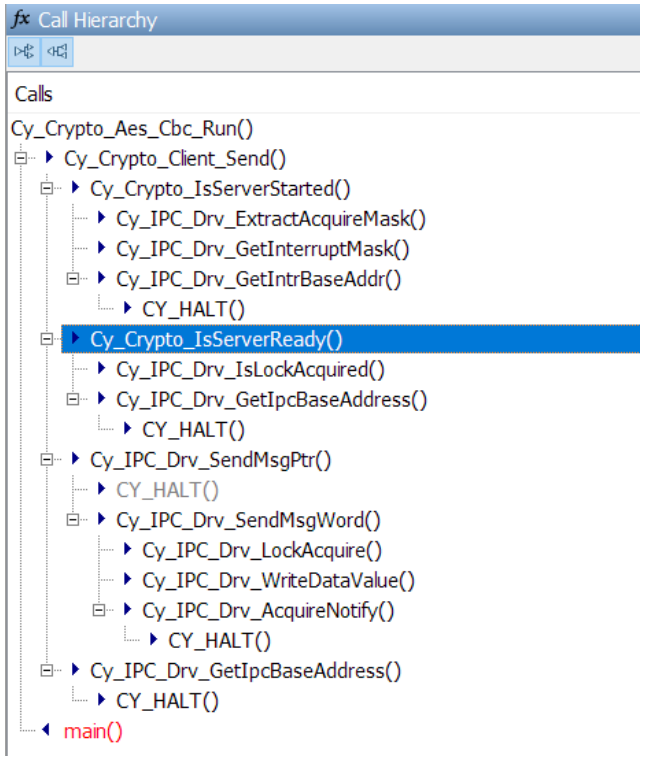
函数调用树



从 Code Prover 提供控制流和数据流分析







问题追溯树

变量访问树

函数调用树

Polyspace 满足认证最高等级要求

ZERTIFIKAT ◆ CERTIFICATE ◆ 認證證書 ◆ СЕРТИФИКАТ ◆ CERTIFICADO ◆ CERTIFICAT

TÜV SÜD Product Service

CERTIFICATE

No. Z10 067052 0018 Rev. 02

Holder of Certificate: The MathWorks, Inc.
3 Apple Hill Drive
Natick MA 01760-2098
USA

Factory(ies): 067779

Certification Mark:

Product: Software Tool for Safety Related Development

Model(s):
Polyspace® Code Prover™
Polyspace® Code Prover™ Server™
Polyspace® Bug Finder™
Polyspace® Bug Finder™ Server™
Polyspace® Client™ and
Polyspace® Server™ for C/C++

Parameters: The verification tools, classified T2, are suitable for use in safety-related development according to IEC 61508 and EN 50128 for any SIL and ISO 25119 for any SRL. The verification tools are qualified tools according to ISO 26262 for any ASIL. They are suitably validated for use in safety-related development according to IEC 62304. The report MN74651C is a mandatory part of this certificate.

Tested according to:
IEC 61508-3:2010
IEC 62304:2015
ISO 25119-3:2018
ISO 26262-8:2018
EN 50128:2011/AC:2014

The product was tested on a voluntary basis and complies with the essential requirements. The certification mark shown above can be affixed on the product. It is not permitted to alter the certification mark in any way. In addition the certification holder must not transfer the certificate to third parties. See also notes overleaf.

Test report no.: MN74651C

Valid until: 2025-01-15

Date, 2020-01-16

(Peter Weiß)

A4 / 07.17

Page 1 of 1
TÜV SÜD Product Service GmbH • Certification Body • Ridlerstraße 65 • 80339 Munich • Germany

TÜV®

Polyspace 满足认证最高等级要求

ZERTIFIKAT ◆ CERTIFICATE ◆ 認證證書 ◆ СЕРТИФИКАТ ◆ CERTIFICADO ◆ CERTIFICAT

TÜV SÜD Product Service

CERTIFICATE

No. Z10 067052 0018 Rev. 02

Holder of Certificate: The MathWorks, Inc.
3 Apple Hill Drive
Natick MA 01760-2098
USA

Factory(ies): 067779

Certification Mark: Functional Safety

Product: Software Tool for Safety Relate

Model(s):
Polyspace® Code Prover™
Polyspace® Code Prover™ Serve
Polyspace® Bug Finder™
Polyspace® Bug Finder™ Server
Polyspace® Client™ and
Polyspace® Server™ for C/C++

Parameter: The verification tools, classified T2, ar… related development according to IE… IL and ISO 25119 for any S… ools according to IS…

Tested according to:
ISO 2…
ISO 26262-8:2…
EN 50128:2011/AC:2014

The product was tested on a voluntary basis and complies with the essential requirements. The certification mark shown above can be affixed on the product. It is not permitted to alter the certification mark in any way. In addition the certification holder must not transfer the certificate to third parties. See also notes overleaf.

Test report no.: MN74651C

Valid until: 2025-01-15

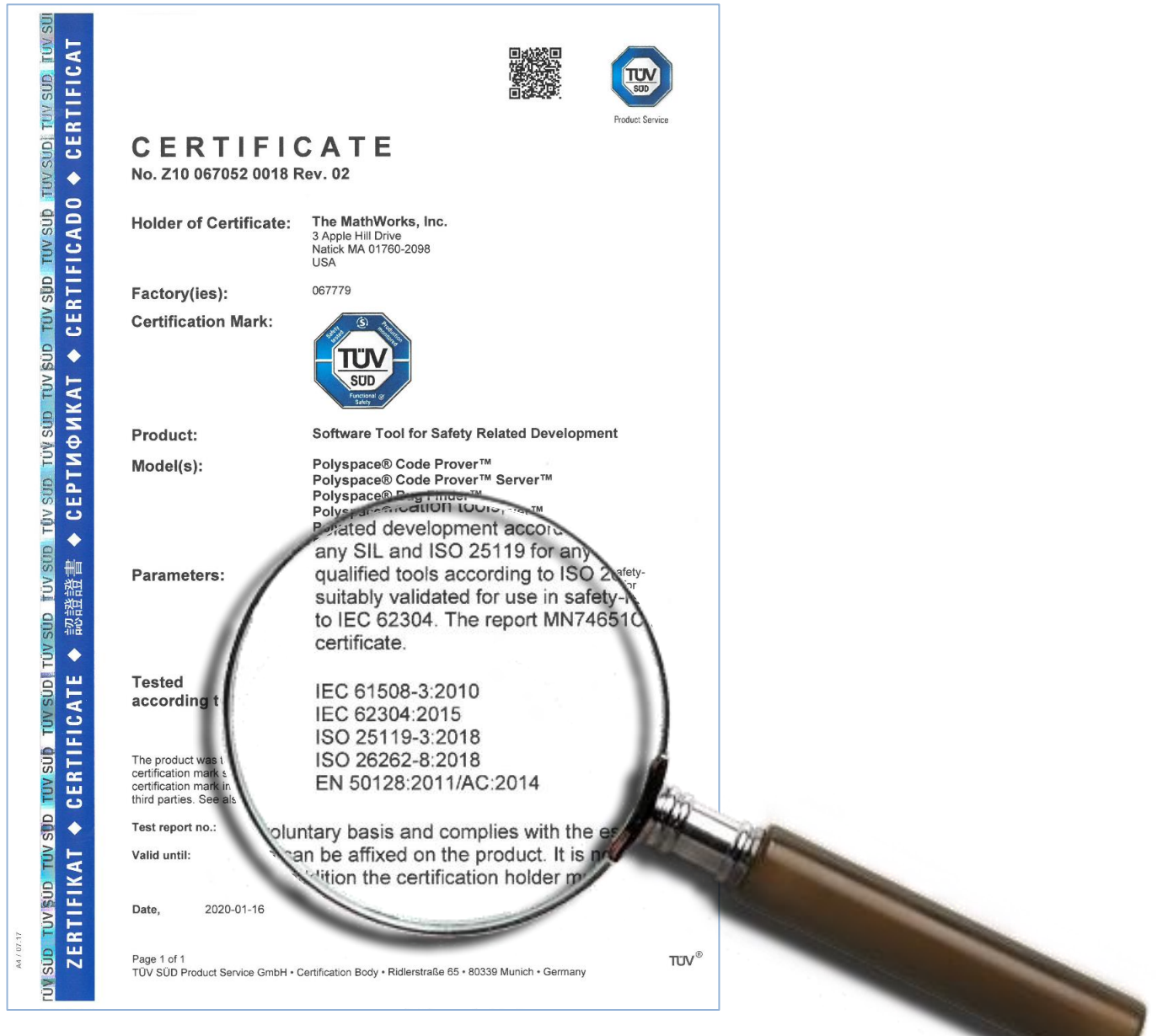
Date, 2020-01-16 (Peter Weiß)

Page 1 of 1
TÜV SÜD Product Service GmbH • Certification Body • Ridlerstraße 65 • 80339 Munich • Germany

TÜV®

A4 / 07.17

Polyspace 满足认证最高等级要求



Polyspace 满足认证最高等级要求


CERTIFICATE
No. Z10 007502 0018 Rev. 02
Holder of Certificate: The MathWorks, Inc.
Factory(ies):
Certification Mark:

TÜV SÜD

Rail

5.7.5 Summary

All Polyspace® tools versions listed in the subsequent table are qualified for all ASILs according to ISO 26262 up to a maximum tool confidence level of TCL2 for Polyspace® Bug Finder™ and TCL3 for Polyspace® Code Prover™. The review of the tool classification and the assessment of the results of the measures applied to qualify the software tools were carried out by TÜV SÜD.

伊必 Elektrobit：使用Polyspace 移除运行时错误
挑战和问题

数千种系统配置

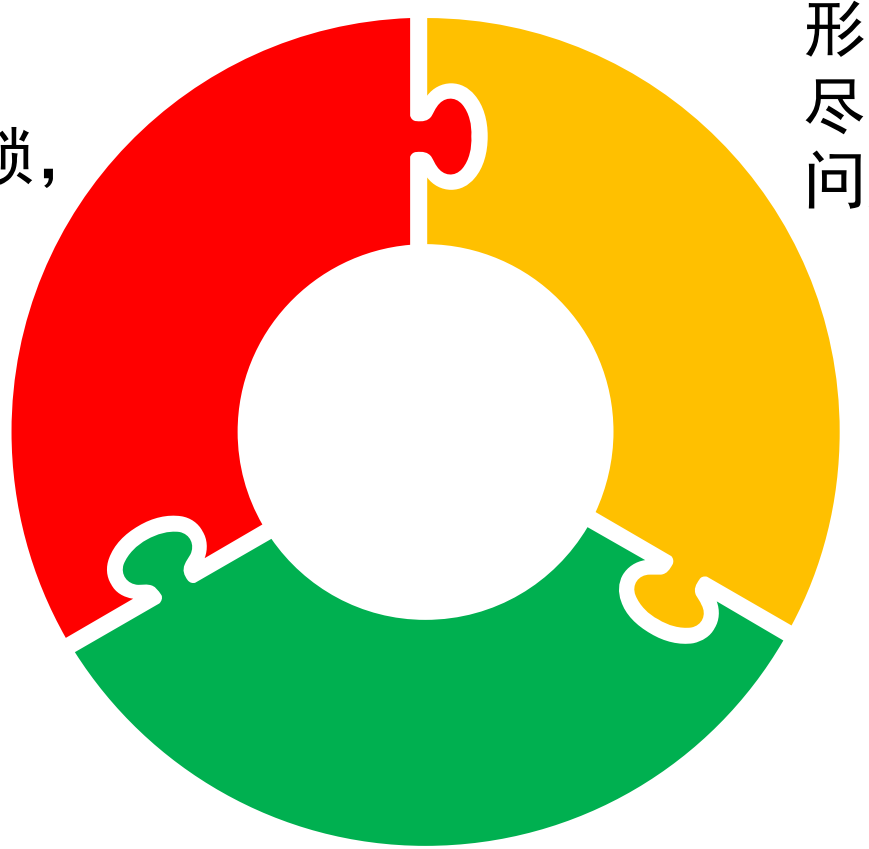
无运行时错误来保
证车辆安全

边界检查
需测试用例和运行

伊必 Elektrobit：使用Polyspace 移除运行时错误

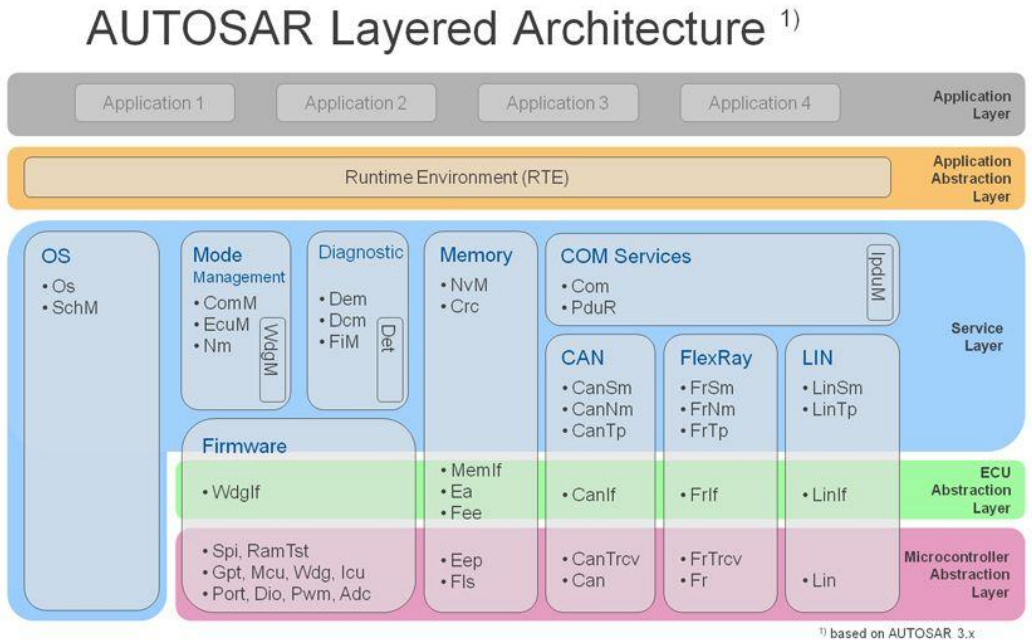
Polyspace 如何解决

减少非必要互斥锁，提高性能

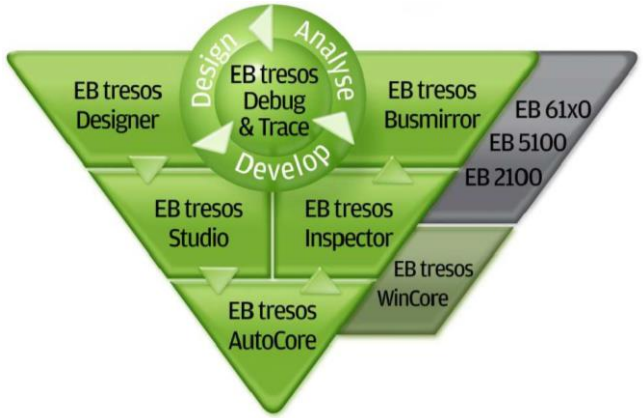


形式化方法穷尽分析，聚焦问题

MISRA C和HIS 代码度量



伊必 Elektrobit：使用Polyspace 移除运行时错误成果总结



- 发现了诸如溢出、变量未初始化、并发等问题
- 代码证明比例超过97%，无 RTE
- 大幅度缩小验证时间
- 通过了ISO 26262 认证
- 使用持续测试技术，每次分析检测

Polyspace code verifiers enable us to demonstrate conclusively that the software we deliver is free of certain run-time errors. More importantly, they enable us to do so faster, more thoroughly, and with less manual review than was previously possible.

案例链接

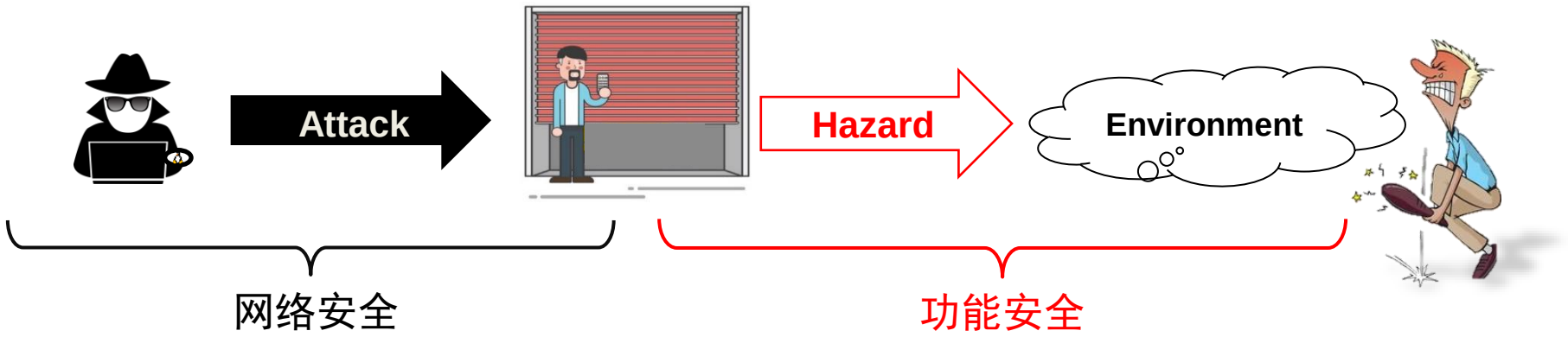
Polyspace 为车联网之网络安全保驾护航

Safety 依赖 Security，Security 促进 Safety

Security: “在受攻击时，保证可靠的运行，Reliable computing in the presence of adversaries”

Safety: “没有危险，Absence of hazard”

网络安全拒绝滥用 功能安全防止危险

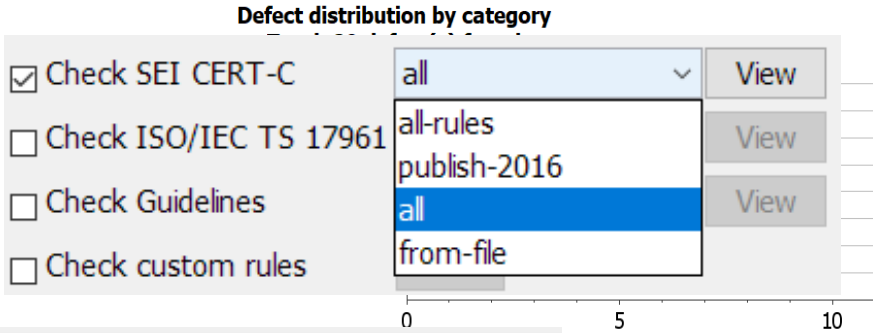


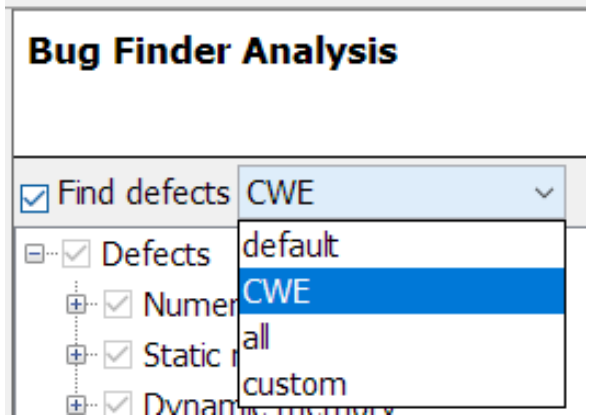
33

使用 Polyspace 进行网络安全检查和证明

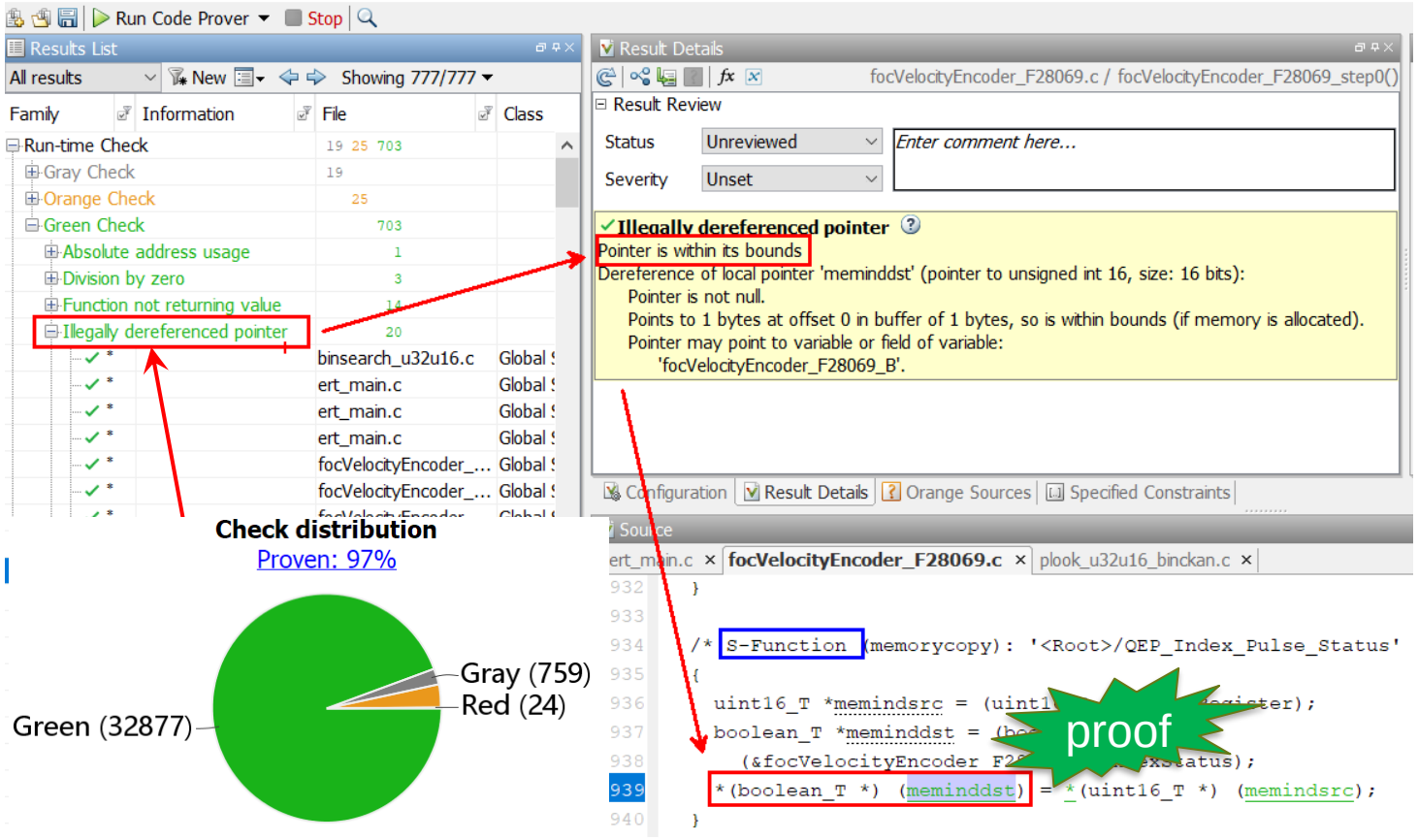
Polyspace Code Verifier

检查安全编码规范和实践指南





证明没有安全漏洞



IEC Certification Kit 提供 ISO/SAE 21434:2021实践指南

IEC Certification Kit

Model-Based Design for ISO/SAE 21434:2021 Road vehicles — Cybersecurity engineering

ISO/SAE 21434:2021 Clause	Applicable Requirements ID and Summary	Applicable Model-Based Design Tools and Processes	Comments
8. Continual cybersecurity activities	[RQ-08-05] Weaknesses evaluation	Polyspace Code Prover	You can use these MathWorks products to analyze system design and implementation for weakness that can contribute to vulnerabilities:
	[RQ-08-07] Vulnerability management	Polyspace Code Prover Server Polyspace Bug Finder Polyspace Bug Finder Server Requirements Toolbox	• Polyspace tools can be used as Static Application Security Testing (SAST) tools to analyze source code for numerous defects related to security weaknesses and vulnerabilities, such as weak cryptographic algorithms. • Requirements Toolbox identifies requirements weakness through the traceability, implementation, and verification status)

形式化方法助力高效运行、低ROM占用的底层软件

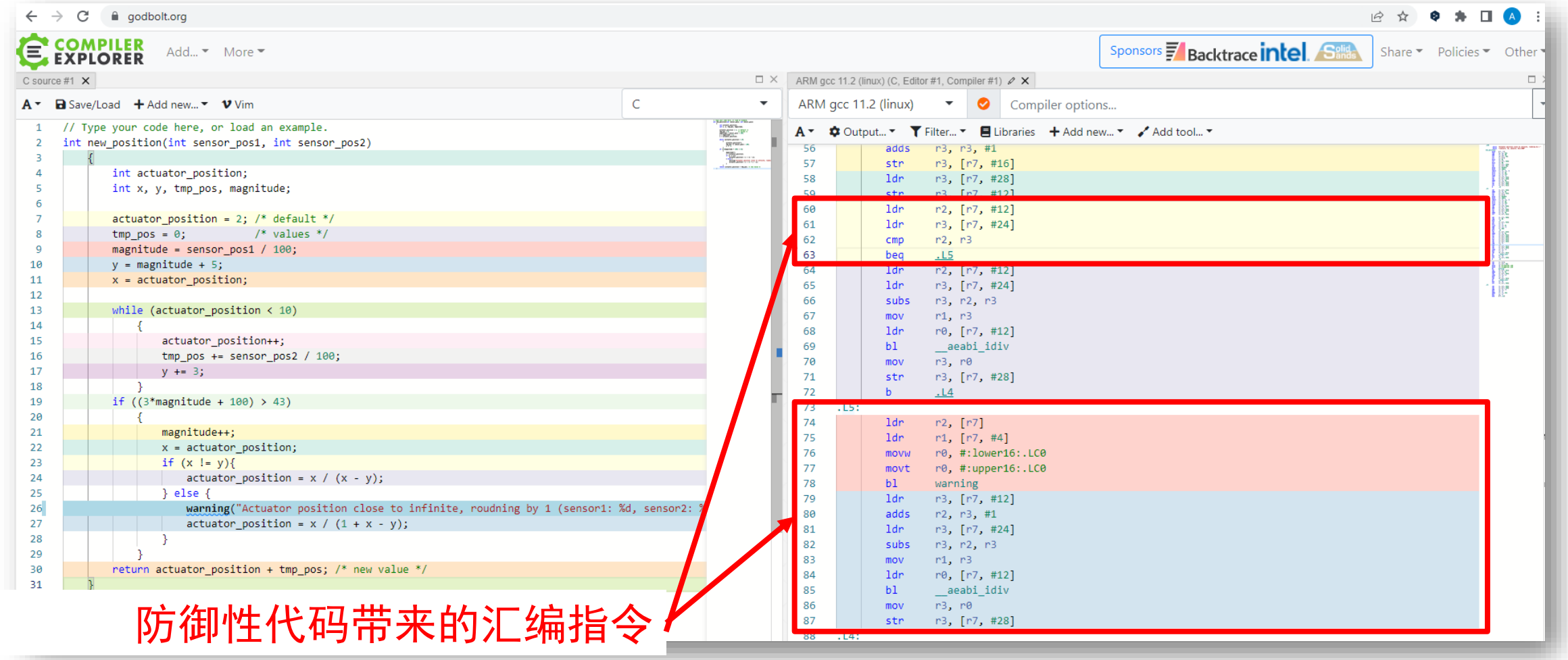
高安全代码常用编码实践：防御性编程

```
3  int new_position_protected(int sensor_pos1, int sensor_pos2)
4  {
5      int actuator_position;
6      int x, y, tmp_pos, magnitude;
7
8      actuator_position = 2; /* default */
9      tmp_pos = 0;           /* values */
10     magnitude = sensor_pos1 / 100;
11     y = magnitude + 5;
12     x = actuator_position;
13
14     while (actuator_position < 10)
15     {
16         actuator_position++;
17         tmp_pos += sensor_pos2 / 100;
18         y += 3;
19     }
20
21     if ((3 * magnitude + 100) > 43)
22     {
23         magnitude++;
24         x = actuator_position;
25         if (x != y)
26         {
27             actuator_position = x / (x - y);
28         }
29         else
30         {
31             warning("Actuator position close to infinite, roudning by 1 (sensor1: %d, sensor2: %d))", sensor_pos1, sensor_pos2);
32             y = magnitude - 3; // update y to value from 1 cycle less
33             actuator_position = x / (x - y);
34         }
35     }
36     return actuator_position + tmp_pos; /* new value */
37 }
```

形式化方法证明，除法安全，无需防御

```
int new_position_protected(int sensor_pos1, int sensor_pos2)
{
        int actuator_position;
        int x, y, tmp_pos, magnitude;

        actuator_position = 2; /* default */
        tmp_pos = 0;           /* values */
        magnitude = sensor_pos1 / 100;
        y = magnitude + 5;
        x = actuator_position;

        while (actuator_position < 10)
        {
                actuator_position++;
                tmp_pos += sensor_pos2 / 100;
                y += 3;
        }

        if ((3 * magnitude + 100) > 43)
        {
                magnitude++;
                x = actuator_position;
                if (x != y)
                {
                        actuator_position = x / (x - y);
                }
                else
                {
                        warning("Actuator position close to infinite, roudning by 1 (sensor1: %d, sensor2: %d))", sensor_pos1, sensor_pos2);
                        y = magnitude - 3; // update y to value from 1 cycle less
                        actuator_position = x / (x - y);
                }
        }
        return actuator_position + tmp_pos; /* new value */
}
```

防御性代码导致资源的浪费



https://godbolt.org/z/G6E7Pfdz4

删除防御性代码

```
int new_position(int sensor_pos1, int sensor_pos2)
{
        int actuator_position;
        int x, y, tmp_pos, magnitude;

        actuator_position = 2; /* default */
        tmp_pos = 0;           /* values */
        magnitude = sensor_pos1 / 100;
        y = magnitude + 5;
        x = actuator_position;

        while (actuator_position < 10)
        {
                actuator_position++;
                tmp_pos += sensor_pos2 / 100;
                y += 3;
        }
        if ((3 * magnitude + 100) > 43)
        {
                magnitude++;
                x = actuator_position;
                actuator_position = x / (x - y);
        }
        return actuator_position + tmp_pos; /* new value */
}
```

- 资源占用更少
- 性能更好
- 维护更容易
- 代码审查更简单

某IC驱动中非必要的条件或防御

```
uint32_t memResult = 1U;

if (size != 0U)
{
        C
                                                    Parameter 'size' (unsigned int 16): 8 or 15 or [19 .. 20] or 28 or 32

                                                    Conversion from unsigned int 16 to unsigned int 32
                                                        right:  8 or 15 or [19 .. 20] or 28 or 32 or 48 or 64
                                                        result: 8 or 15 or [19 .. 20] or 28 or 32 or 48 or 64

                                                                                                    Press 'F2' for focus
    /* I
    Crypto_Run3ParamInstr(base,
                          CRYPTO_V1_STR_MEMCMP_OPC,
                          CRYPTO_RSRC0_SHIFT,
                          CRYPTO_RSRC4_SHIFT,
                          CRYPTO_RSRC8_SHIFT);
```

多余的判定条件

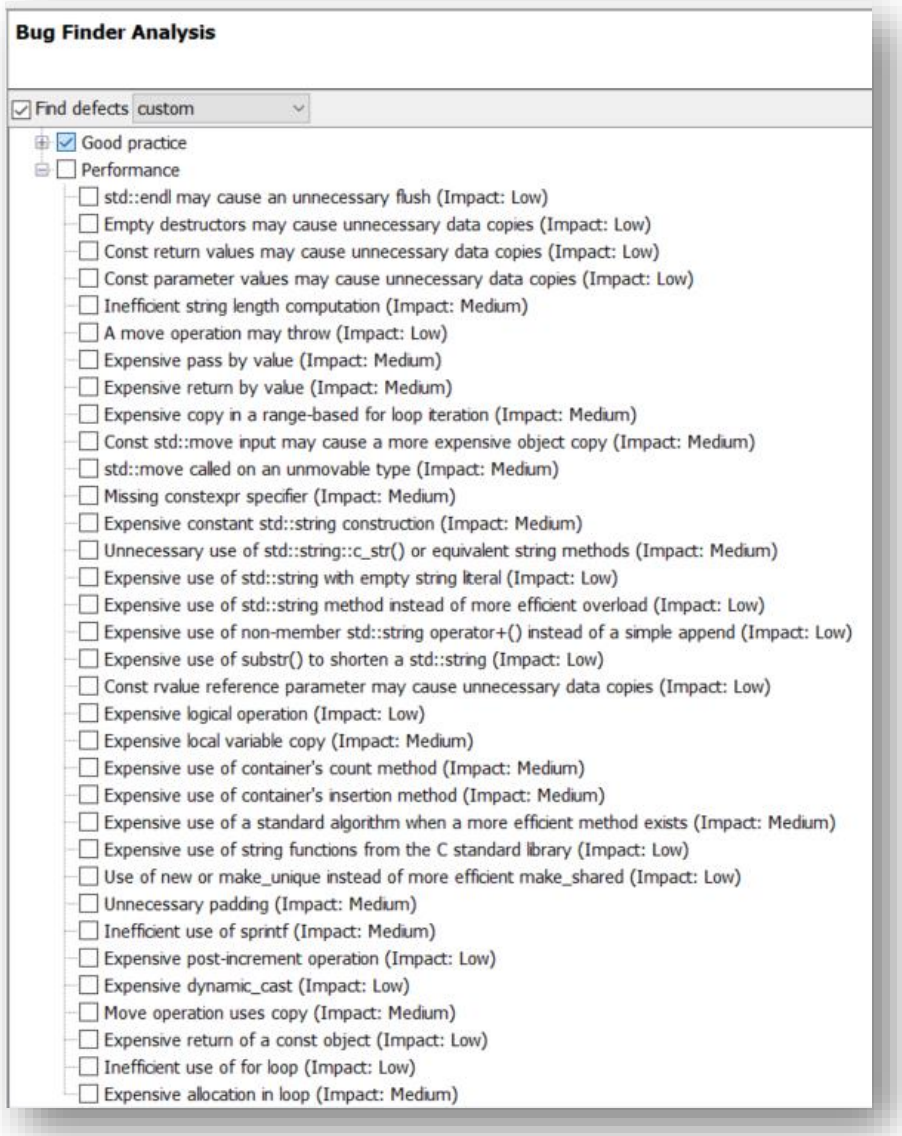
```
switch (mul_red_alg_select)
{
    case     CRYPTO_NIST_P_                                  Static variable 'mul_red_alg_select' (unsigned enum 32): 0 or 2
                                                                                                    Press 'F2' for focus
             /* Curve-specific multiplication reduction algorithms */
                Crypto_Core_EC_CS_MulRed(base, z, x, size);
        break;
    case     CRYPTO_NIST_P_SHIFT_MUL_RED_ALG:
             /* Shift-multiply, curve-specific multiplication reduction algorithms */
                Crypto_Core_EC_SM_MulRed(base, z, x, size);
        break;
    default:
             /* Generic Barrett modular reduction */
                Crypto_Core_EC_Bar_MulRed(base, z, x, size);
        break;
```

不可用的分支

```
/* Verify operation result */
if (timeout > 0U)
{
                                                    Local variable 'timeout' (unsigned int 32): 0
                                                                                                    Press 'F2' for focus
        USBF                                                                    tion, endpoint);
    retStatus =     USBFS_DEV_DRV_SUCCESS;
}
```

多余的防御操作

更多关于软件性能的检查项



Expensive allocation in loop (Impact: Medium)
malloc and *free* called in a loop with a constant buffer size.
Moving these calls outside the loop will avoid unnecessary work.

	Event	File	Scope	L...
1	This constant expression is passed to *malloc*	expensive_by_value.c	expensive_by_value.c	30
2	This *free* call releases a constant amount of memory every loop iteration	expensive_by_value.c	expensive_by_value.c	32
3	Expensive allocation in loop	expensive_by_value.c	Loop()	30

Configuration | Result Details

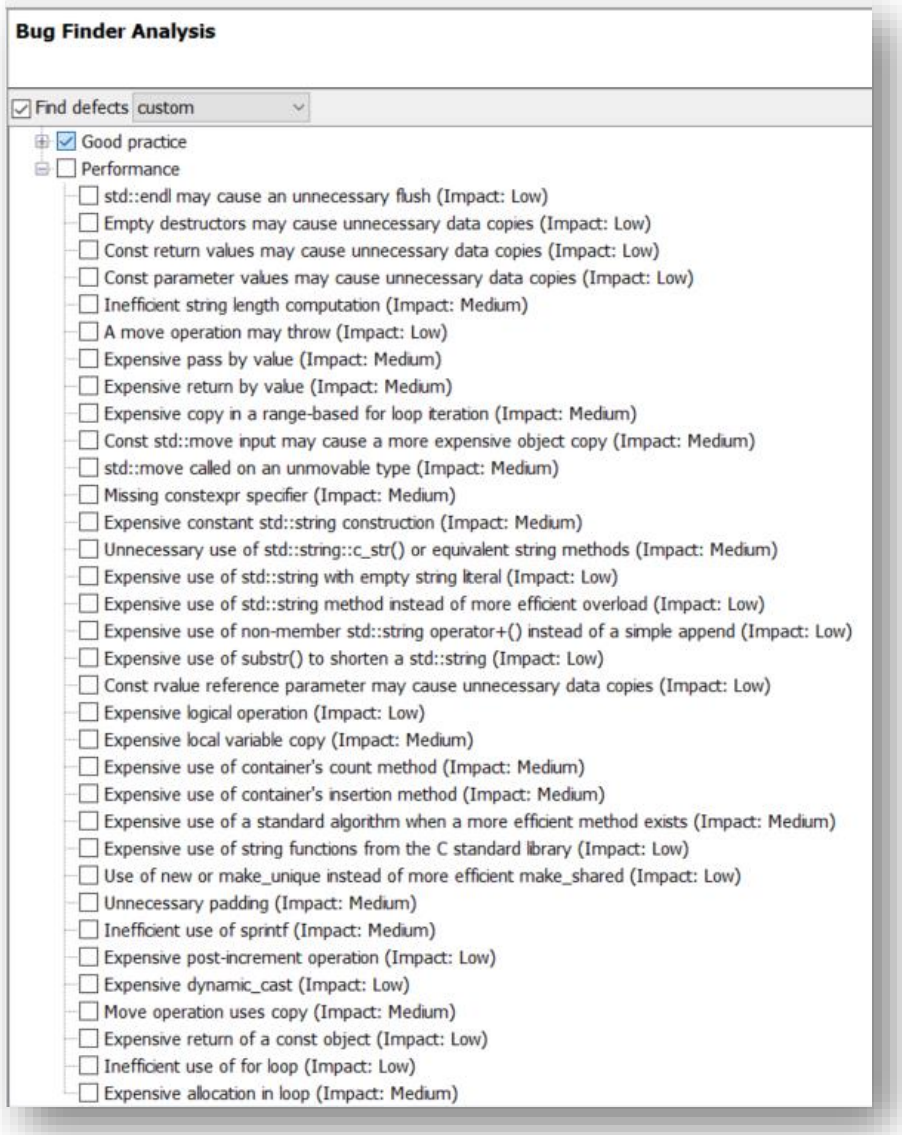
Source

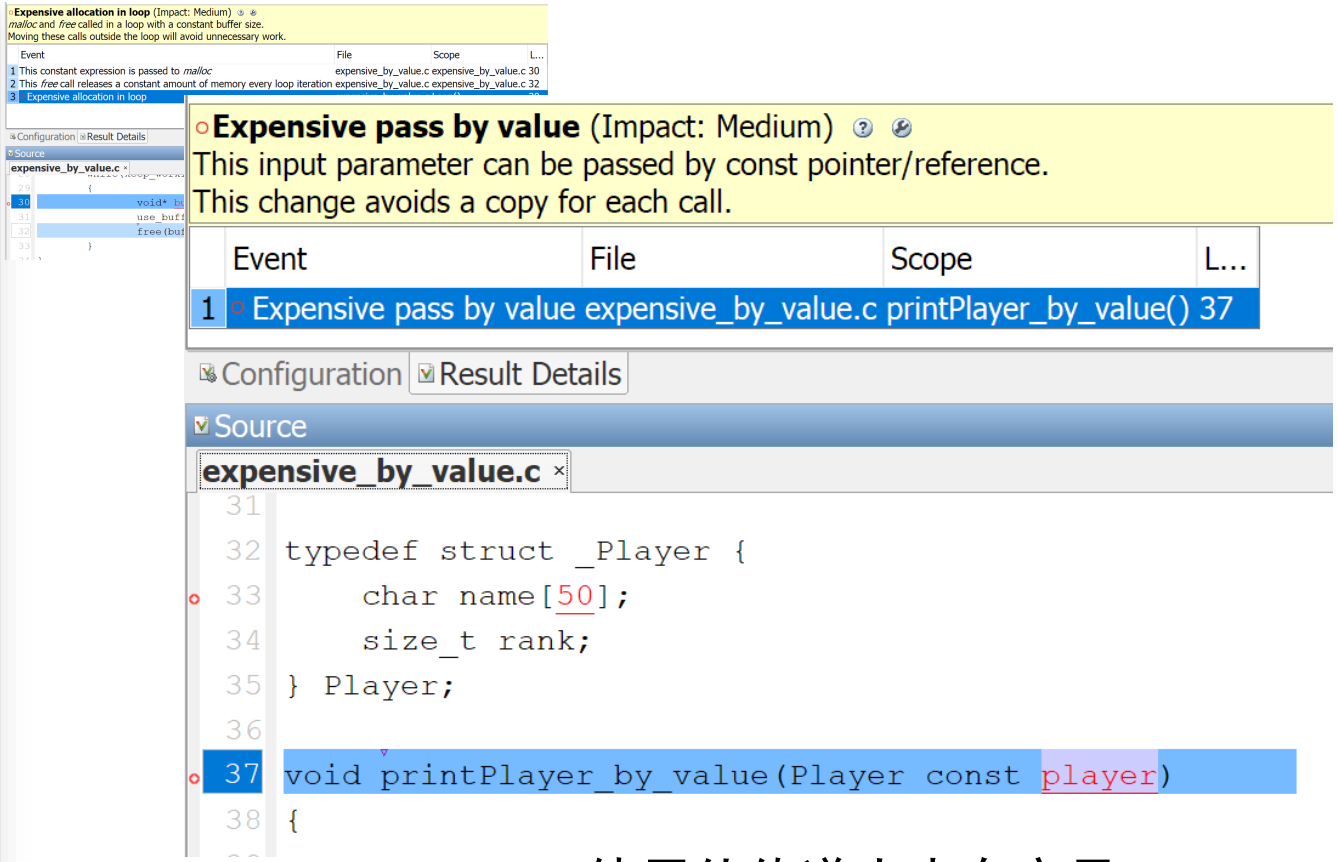
expensive_by_value.c

```
29          {
30              void* buffer = malloc(10); // Defect
31              use_buffer(buffer);
32              free(buffer);
33          }
```

在循环中分配内存

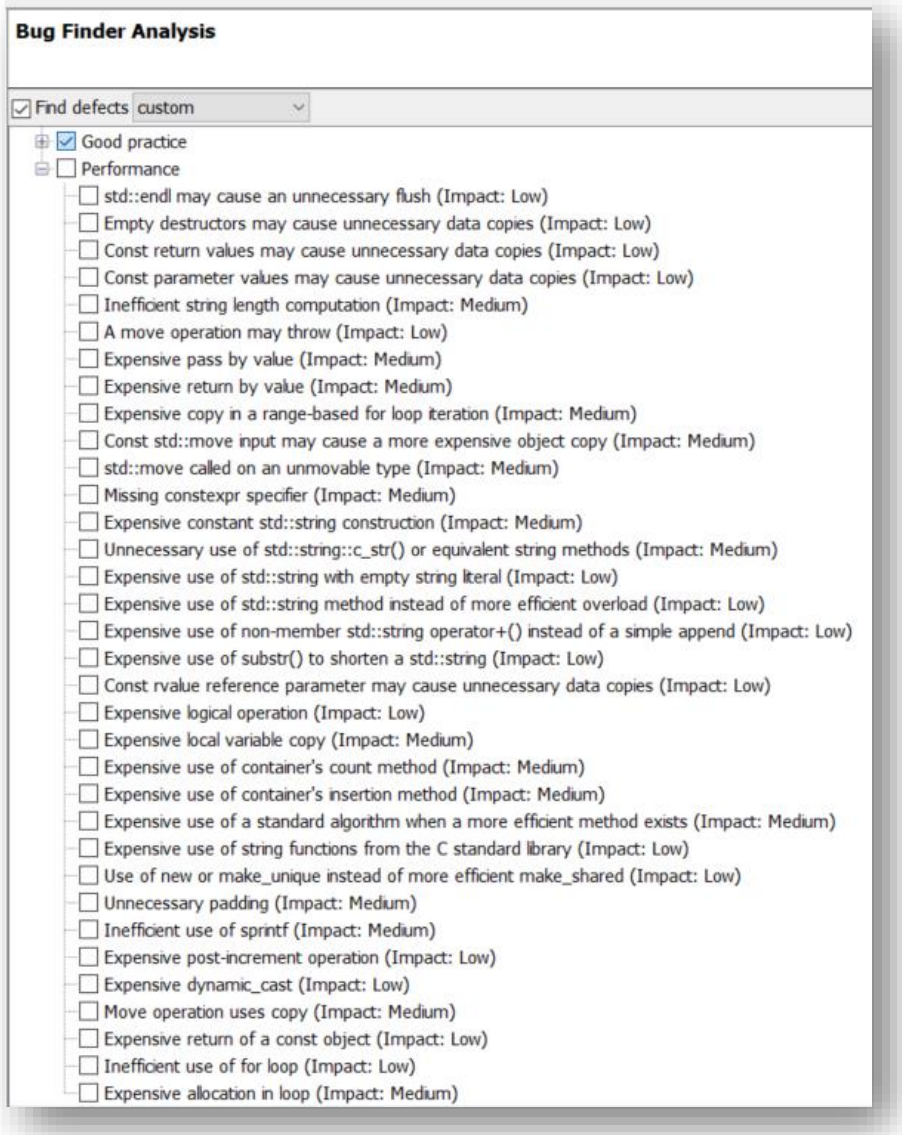
更多关于软件性能的检查项

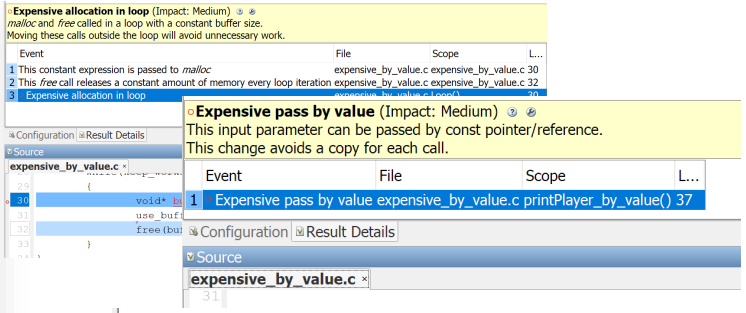


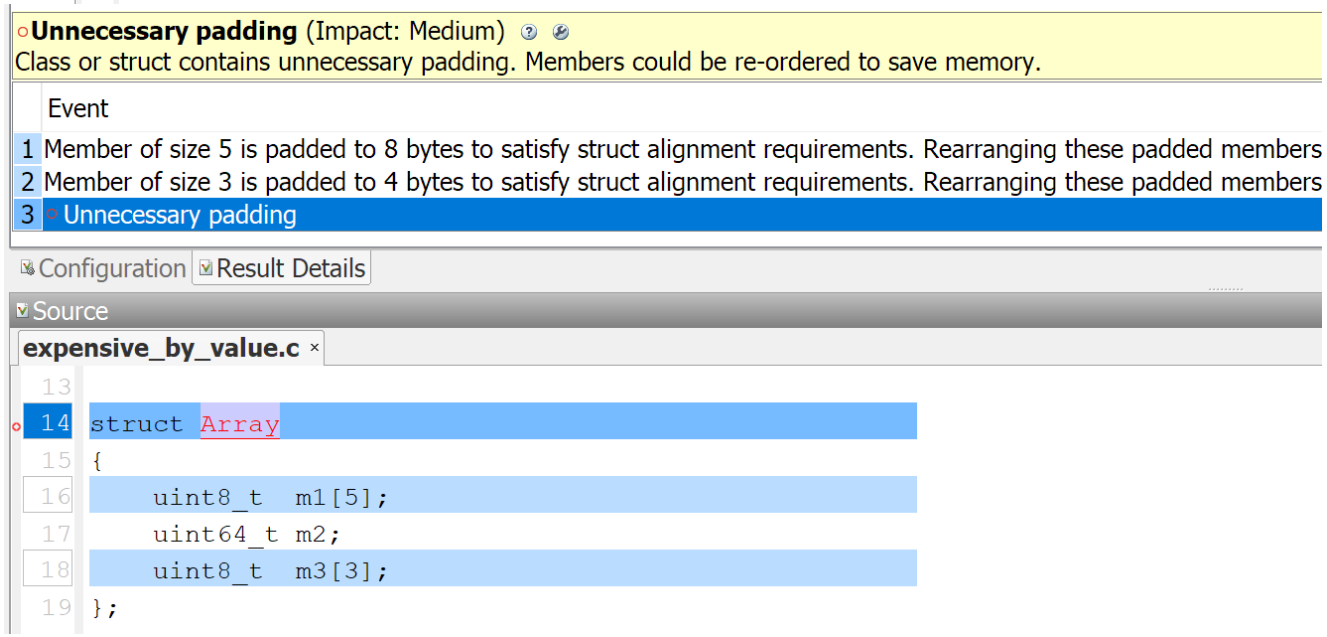


使用值传递大内存变量

更多关于软件性能的检查项



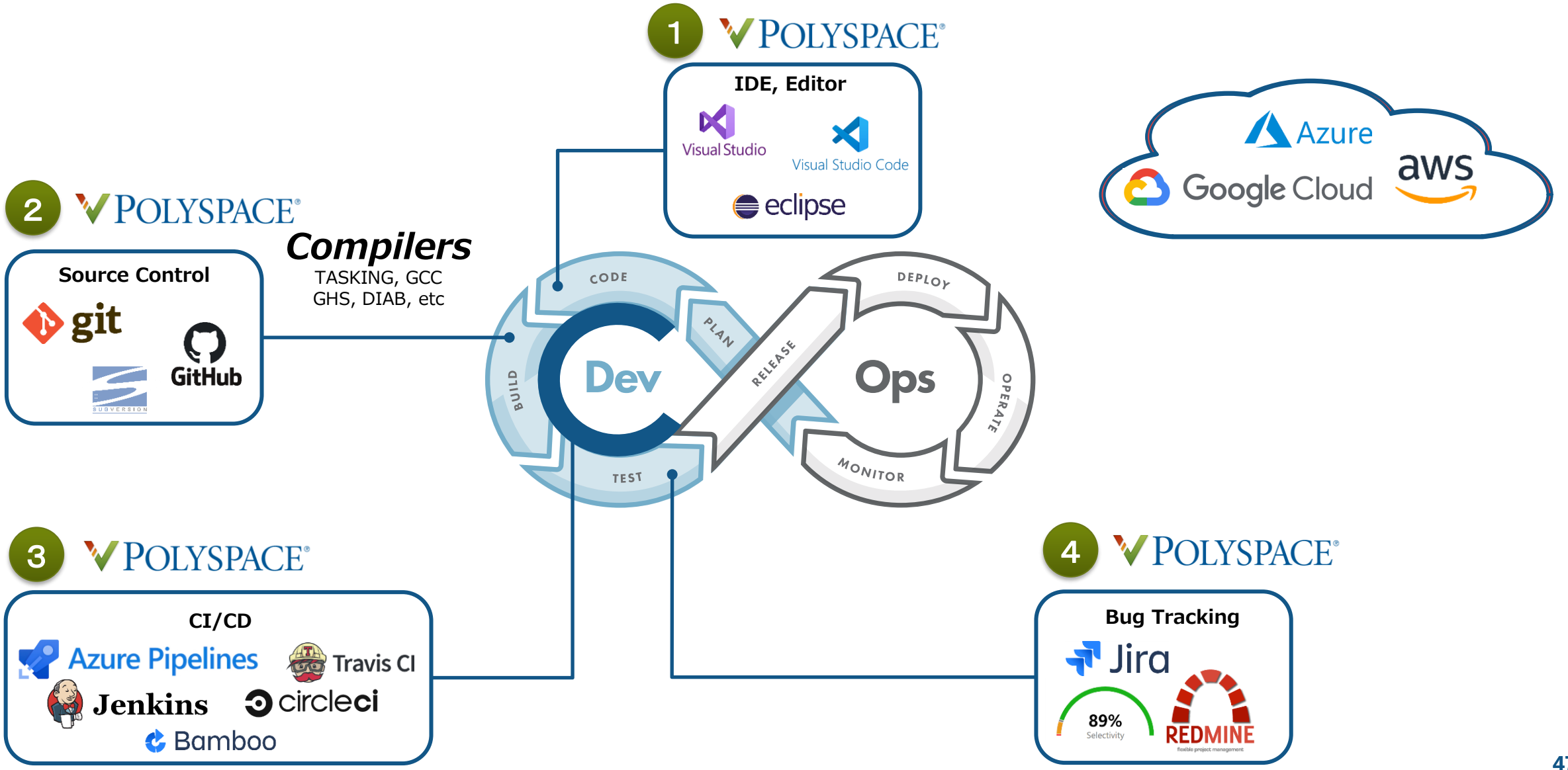




不合理的结构体成员顺序增加内存使用

将 Polyspace 运用到开发流程中

集成到 DevOps 中


1
POLYSPACE
IDE, Editor
Visual Studio
Visual Studio Code
eclipse
2
POLYSPACE
Source Control
git
GitHub
SUBVERSION
Compilers
TASKING, GCC
GHS, DIAB, etc
CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR
Dev
Ops
Azure
Google Cloud
aws
3
POLYSPACE
CI/CD
Azure Pipelines
Travis CI
Jenkins
circleci
Bamboo
4
POLYSPACE
Bug Tracking
Jira
89%
Selectivity
REDMINE
flexible project management

集成到 DevOps 中

1 POLYSPACE

IDE, Editor

Visual Studio

Visual Studio Code

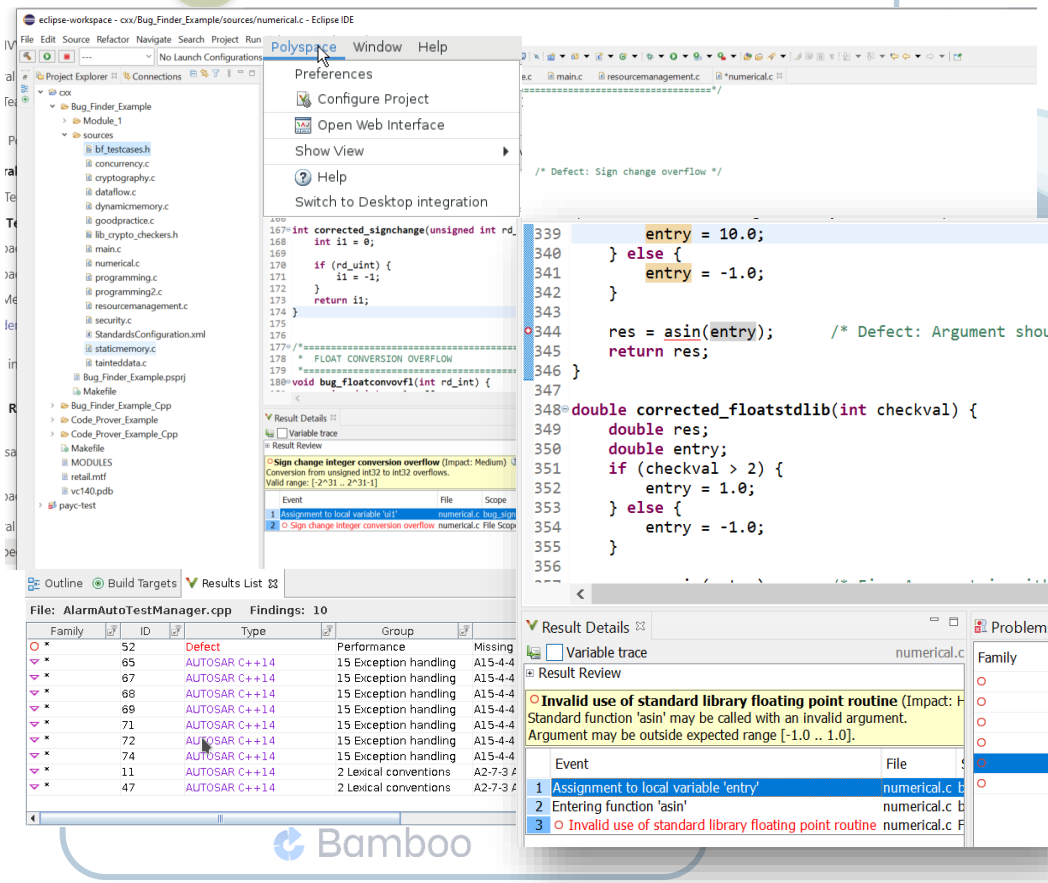
eclipse

Azure

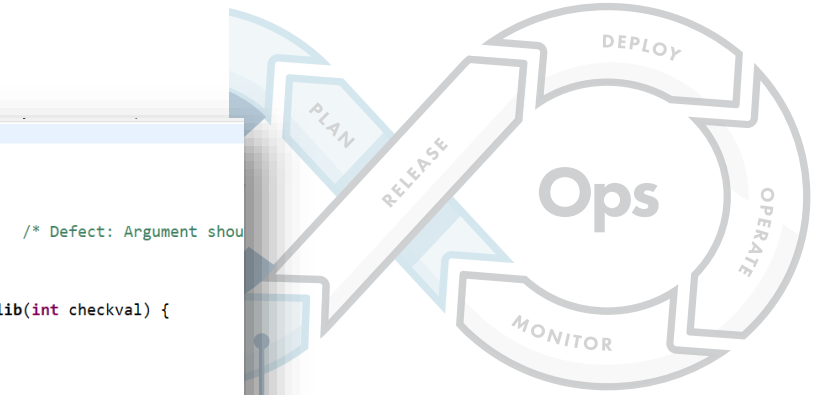
Google Cloud

aws

2 POLYSPACE



Bamboo



4 POLYSPACE

Bug Tracking

Jira

89%
Selectivity

REDMINE

集成到 DevOps 中

```
polyspace-configure -allow-build-error -allow-overwrite -module -output-options-path options_path make -B

clude/ src/idl_options.proto

-DBTHREAD_USE_FAST_PTHREAD_MUTEX -D__const__= -D_GNU_SOURCE -DUSE_SYMBOLIZE -DNO_TCMALLOC -D__STDC_FORMAT_MACROS -D_
35\|2022-04-27T10:25:52+08:00\" -O2 -pipe -Wall -W -fPIC -fstrict-aliasing -Wno-invalid-offsetof -Wno-unused-paramete
r.cpp -o src/mcpack2pb/generator.o
```

```
y/esp_authenticator.o src/brpc/policy/file_naming_service.o src/brpc/policy/hasher.o src/brpc/policy/consul_naming_service.o src/
c/brpc/policy/redis_protocol.o src/brpc/policy/ubrpc2pb_protocol.o src/brpc/policy/auto_concurrency_limiter.o -Xlinker "-)" -lgfla
t
> Copying to output/include
> Copying to output/lib
> Copying to output/bin
polyspace-configure: 416s: WARNING: Compilation units detected during your build in module libbrpc_a use different languages.
polyspace-configure: 416s: WARNING: Compilation units detected during your build in module libbrpc_so use different languages.
```

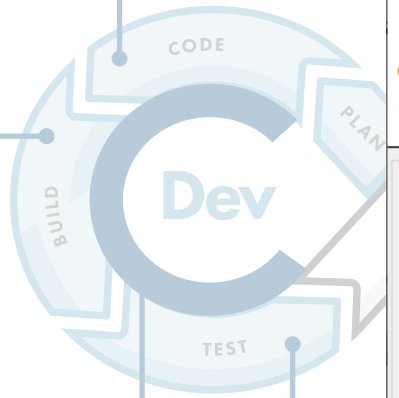
2 POLYSPACE

Source Control

git GitHub SUBVERSION

Compilers

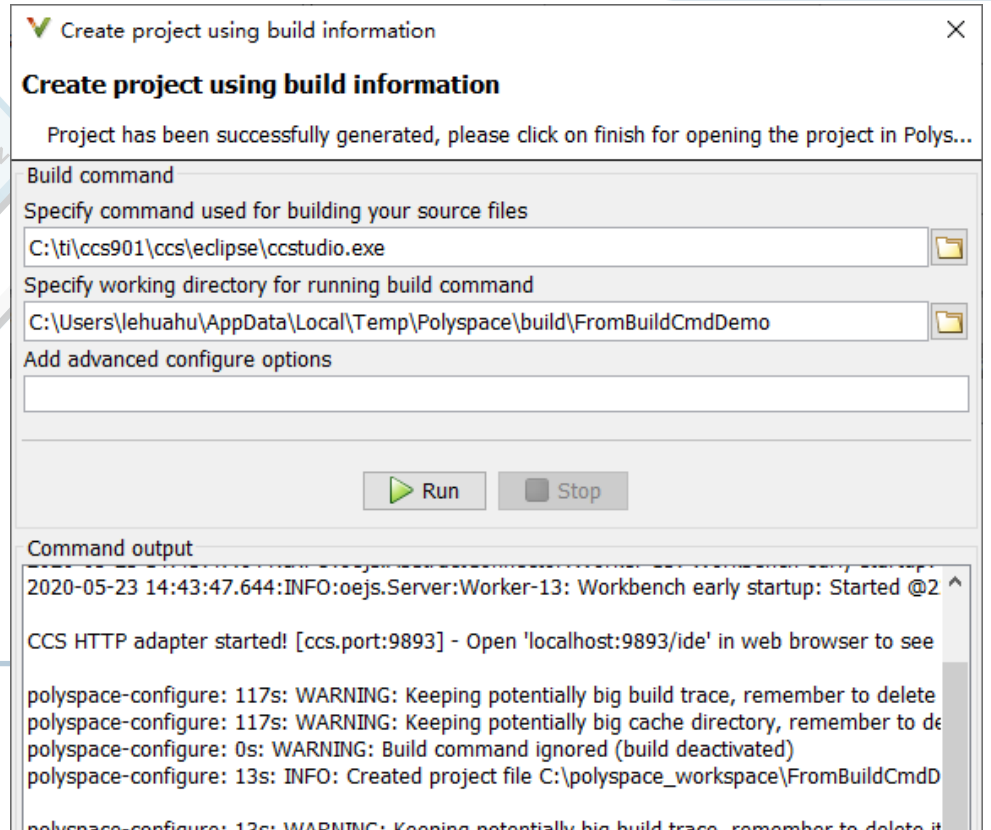
TASKING, GCC
GHS, DIAB, etc



3 POLYSPACE

CI/CD

Azure Pipelines Travis CI Jenkins circleci Bamboo



集成到 DevOps 中

2 POLYSPACE

Source Control

git GitHub SUBVERSION

Compilers

TASKING, GCC
GHS, DIAB, etc

3 POLYSPACE

CI/CD

Azure Pipelines Travis CI Jenkins circleci Bamboo

流水线

定义

Pipeline script

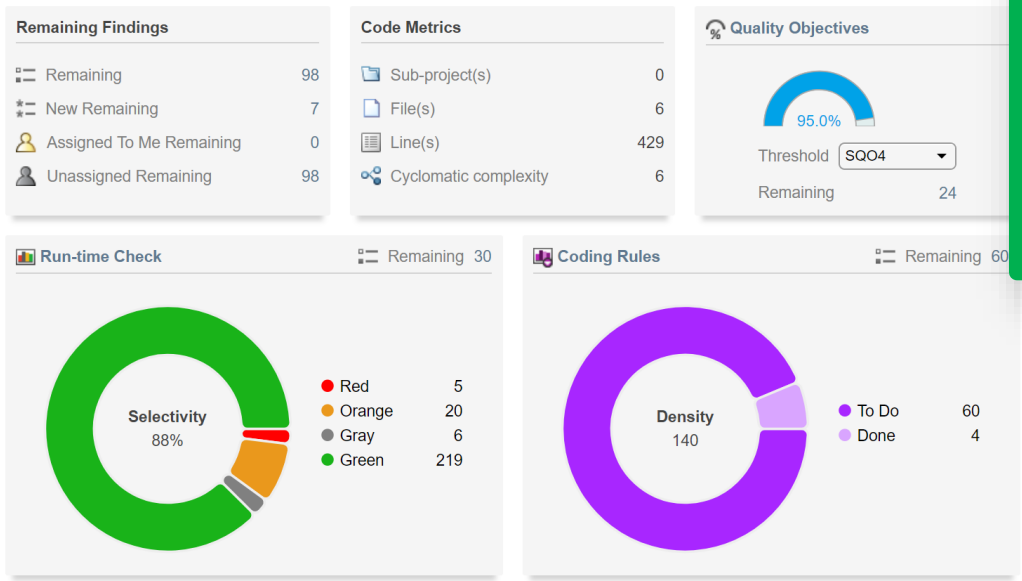
脚本 ?

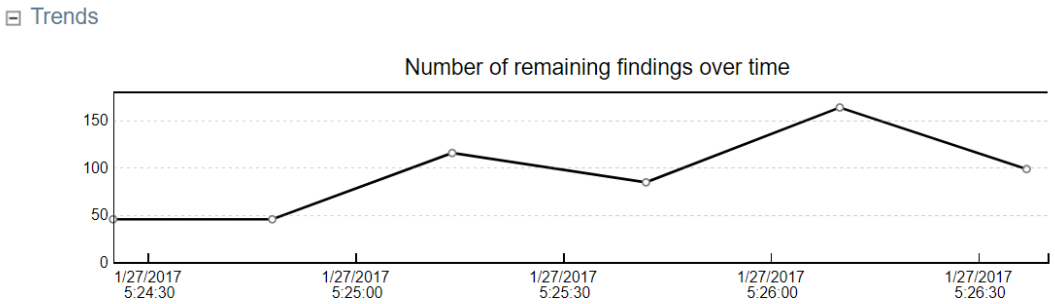
```
59          {
60              sh label: "Cleanup", script: "make clean"
61              sh label: "Polyspace configure", script: "$configure -allow-build-error -allow-overwrite -lang
62          }
63          stage ("Analyze")
64          {
65              sh label: "Polyspace analysis",
66              script: "$analyze -options-file ${PROG}.opts -misra3 all -code-metrics -results-dir $RESULT"
67          }
68
69      // remove this paragraph if you do not use Polyspace Access
```

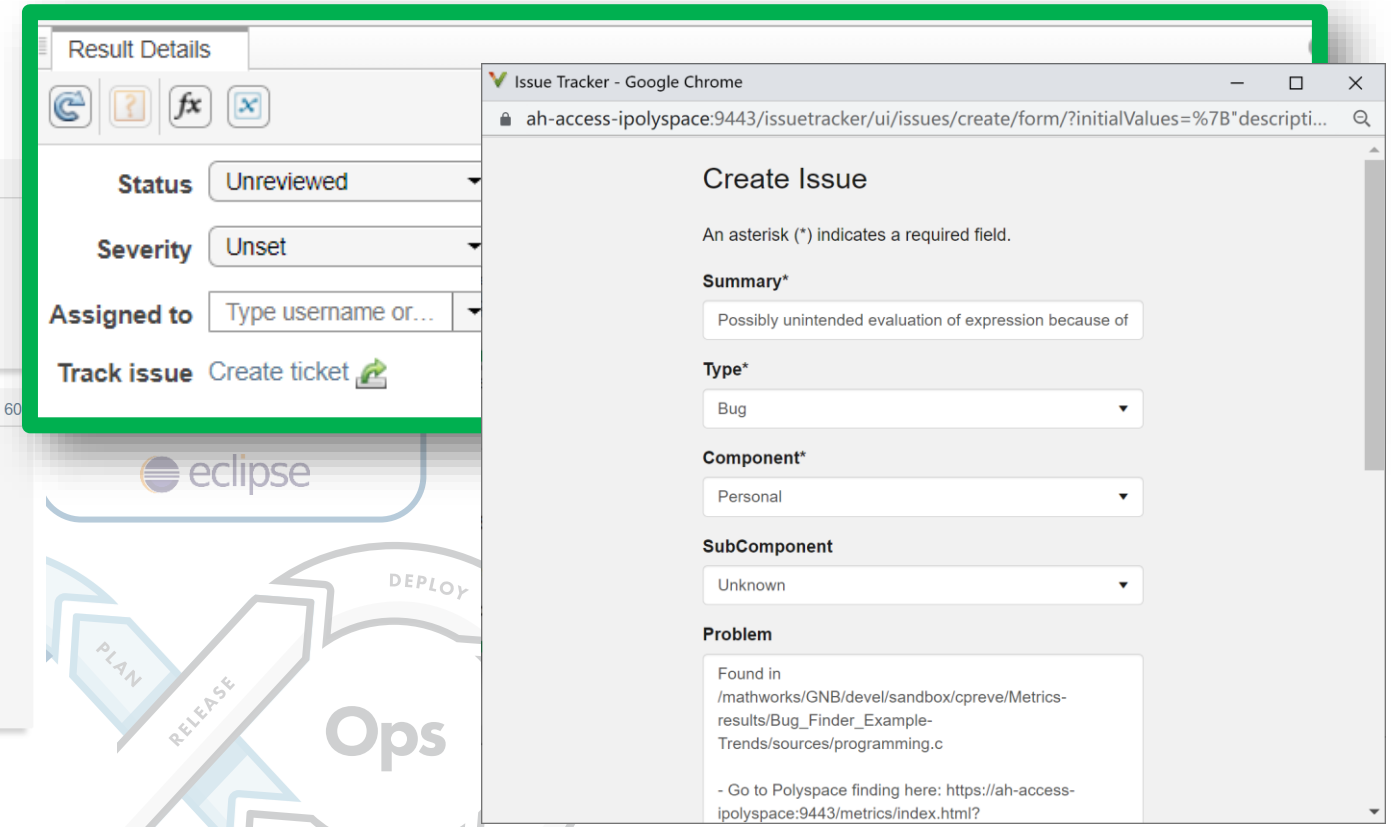
阶段视图

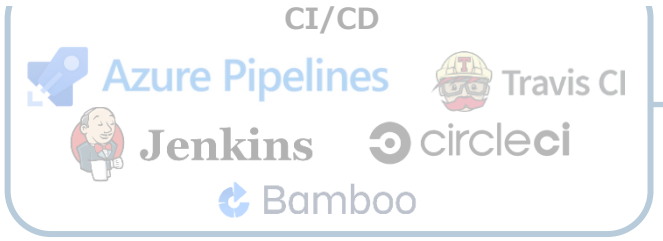
	Prepare	Checkout	Build	Polyspace Analysis	Upload to Access	Quality Gate	Email Notification
Average stage times: (Average full run time: ~40s)	374ms	2s	9s	12s	4s	8s	875ms
#41 Sep 13 15:35 No Changes	323ms	2s	10s	13s	4s	8s	1s
#40 Sep 10 23:08 No Changes	303ms	1s	9s	11s	5s	8s	1s
#39 Sep 10 22:52 No Changes	471ms	3s	10s	15s	4s	8s	1s

集成到 DevOps 中

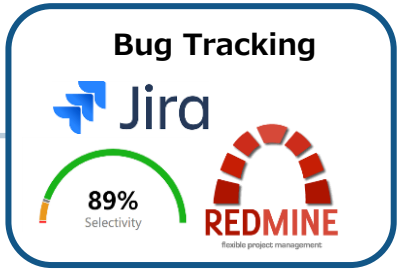

Remaining Findings
Remaining 98
New Remaining 7
Assigned To Me Remaining 0
Unassigned Remaining 98
Code Metrics
Sub-project(s) 0
File(s) 6
Line(s) 429
Cyclomatic complexity 6
Quality Objectives
95.0%
Threshold SQO4
Remaining 24
Run-time Check
Remaining 30
Selectivity
88%
Red 5
Orange 20
Gray 6
Green 219
Coding Rules
Remaining 60
Density
140
To Do 60
Done 4


Trends
Number of remaining findings over time
150
100
50
0
1/27/2017 5:24:30
1/27/2017 5:25:00
1/27/2017 5:25:30
1/27/2017 5:26:00
1/27/2017 5:26:30


Result Details
Status Unreviewed
Severity Unset
Assigned to Type username or...
Track issue Create ticket
eclipse
Issue Tracker - Google Chrome
ah-access-ipolyspace:9443/issuetracker/ui/issues/create/form/?initialValues=%7B"descripti...
Create Issue
An asterisk (*) indicates a required field.
Summary*
Possibly unintended evaluation of expression because of
Type*
Bug
Component*
Personal
SubComponent
Unknown
Problem
Found in /mathworks/GNB/devel/sandbox/cpreve/Metrics-results/Bug_Finder_Example-Trends/sources/programming.c
- Go to Polyspace finding here: https://ah-access-ipolyspace:9443/metrics/index.html?
DEPLOY
PLAN
RELEASE
Ops
MONITOR


CI/CD
Azure Pipelines
Travis CI
Jenkins
circleci
Bamboo


4
POLYSPACE


Bug Tracking
Jira
89%
Selectivity
REDMINE

要点回顾

- 形式化方法及抽象解释法
- Polyspace对 ISO 26262 和 ISO/SAE 21434 支持
- 形式化方法验证用于性能提升
- 将 Polyspace 应用到开发流程中

MATLAB EXPO

感谢您的聆听！

MathWorks®