



5月 - 北京、上海、深圳  
**2014 MATLAB**  
**巡回研讨会**

迎接复杂性设计的挑战



# 使用Polyspace进行软件代码运行错误检查和验证

李春彦

应用工程师

MathWorks China

# 内容

- Polyspace背景介绍
- Polyspace产品功能及使用技巧
- 在基于模型设计开发流程中使用Polyspace
- Polyspace及相关工具对认证的支持

# Hello, Polyspace!

1996



Rain Deutsch

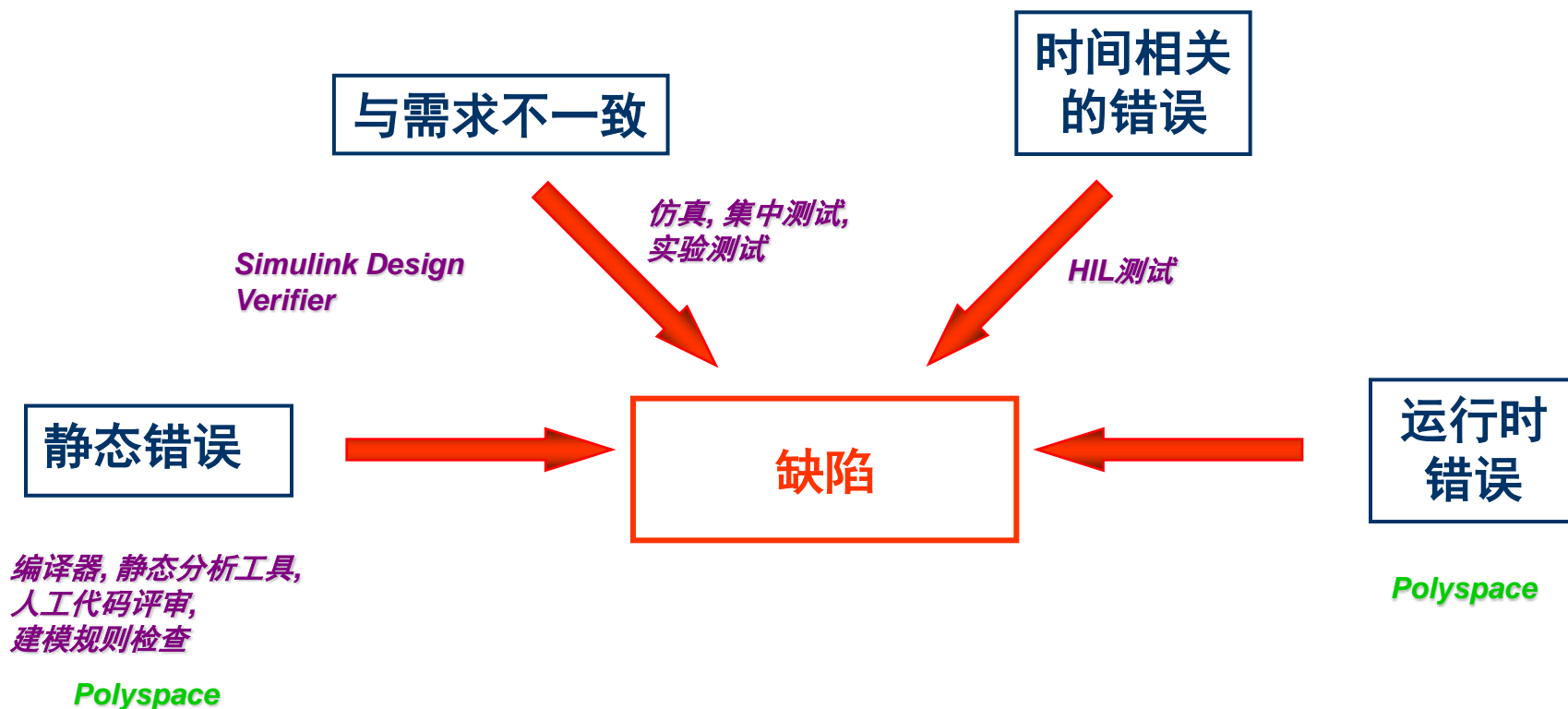
 **MathWorks®** 收购Polyspace

# Polyspace: 静态代码分析工具



- 作用：解决软件的鲁棒性问题，确保软件的安全性和可靠性；
- 能够做什么？
  - 发现编码错误
  - 检查编码规则一致性（MISRA/JSF）
  - 提供代码静态度量信息（圈复杂度等）
  - 证明代码中确定存在或者确定不存在的错误
  - 表示代码已经达到了预期的质量水平
  - 认证帮助：DO-178 C, ISO 26262, ...

# 嵌入式系统的缺陷



什么样的工具和流程能够发现这些缺陷?  
你能证明系统不存在特定缺陷吗?

# 常见的软件代码错误

- 运行时错误
  - 40%左右的软件错误为运行时错误；
  - 运行时错误是软件代码的潜在缺陷，并不频繁出现；
  - 运行期错误会导致系统崩溃和意外的行为。
- 并发问题
- 编程错误
- 不可达代码
- 静态或者动态内存错误

# Polyspace C / C ++产品系列



- Polyspace Bug Finder
  - 快速找到嵌入式软件中的错误
  - 检查代码符合MISRA和JSF编码规则
  - 供软件工程师日常使用
- Polyspace Code Prover
  - 证明代码是安全和可靠的(是否包含运行时错误)
  - 软件组件的深层验证
  - 执行QA对生产就绪代码的签收

还支持Ada语言代码证明

# 使用Polyspace迎接复杂系统设计的挑战



现代汽车动力系统  
500-1,000(KLOC)



波音787飞控系统  
6,500 (KLOC)



航天器\*  
3-1,700 (KLOC)

- 减少软件故障风险
  - 应用于质量关键、业务关键或者高完整性系统中；
  - 减少产品代码中可能导致系统崩溃的潜在错误。
- 缩短软件测试和验证周期
  - 减少软件的鲁棒性测试；
  - 显著缩短系统开发周期。

# 内容

- Polyspace背景介绍
- Polyspace产品功能及使用技巧
- 在基于模型设计开发流程中使用Polyspace
- Polyspace及相关工具对认证的支持

## 如何找到代码的运行时错误？

```
0 k=ioread_i32 ();
1 I=2;
2 J=K+5;
3 while (I<10) {
4     I=I+1;
5     J=J+3;
6 }
7
8 ... / (I-J);    当k=-19时，第8行会出现除零错误
```

第8行有没有潜在的“除零”风险？

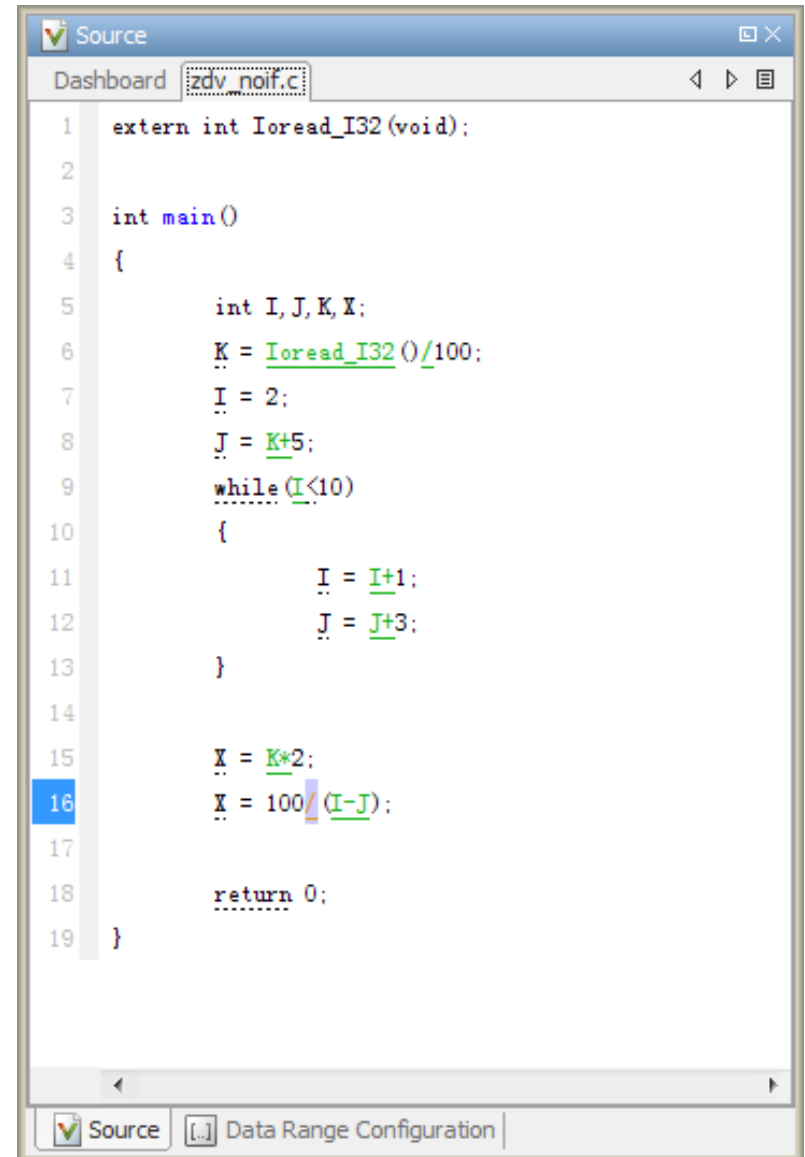
# Polyspace是如何显示的？

```

0   k=ioread_i32();
1   I=2;
2   J=K+5;
3   while (I<10) {
4       I=I+1;
5       J=J+3;
6   }
7
8   ... / (I-J);

```

Polyspace

```

Source
Dashboard  zdv_noif.c
1   extern int Ioread_I32(void);
2
3   int main()
4   {
5       int I, J, K, X;
6       K = Ioread_I32()/100;
7       I = 2;
8       J = K+5;
9       while (I<10)
10      {
11          I = I+1;
12          J = J+3;
13      }
14
15      X = K*2;
16      X = 100/(I-J);
17
18      return 0;
19  }

```

# Polyspace的结果

A colorful world

Proven

**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++;
        }
    }

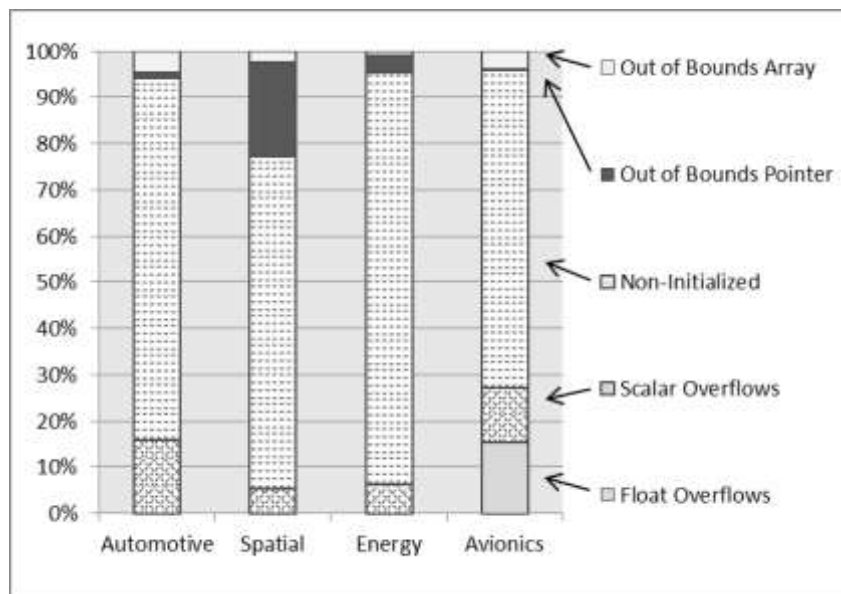
    i = get_bus_status();

    if (i >= 0) {
        *(p - i) = 10;
    }
}
```

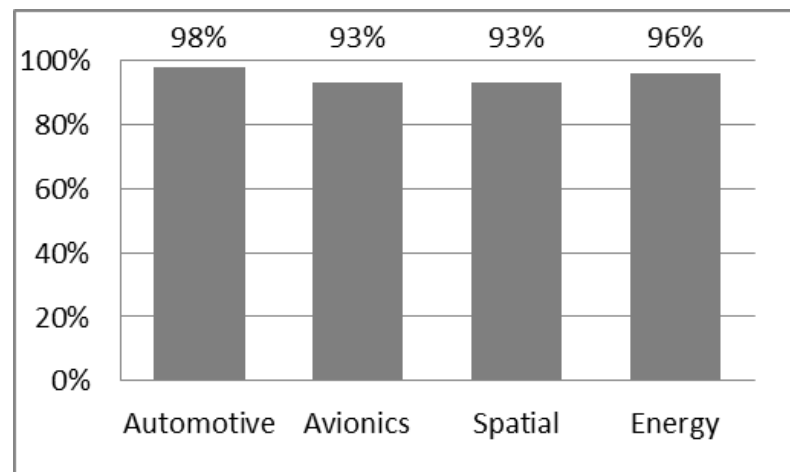
variable 'i' (int32): [0 .. 99]  
assignment of 'i' (int32): [1 .. 100]

# Polyspace行业应用

- Polyspace对于任何类型的代码都是高效的



不同应用的代码缺陷分布



不同应用的代码证明比例

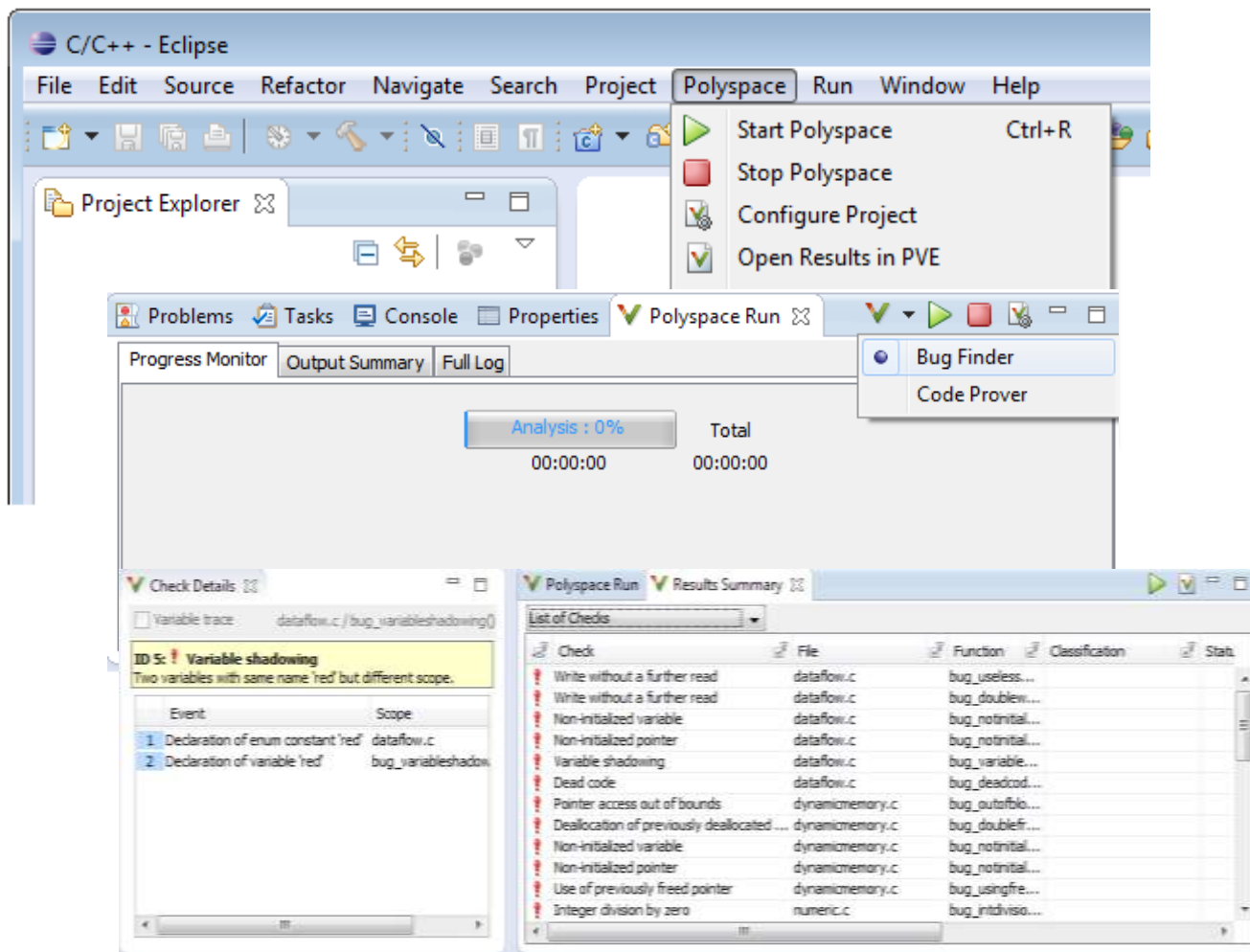
98%意味着只有2%的检查是橙色的

Polyspace采用的形式化方法是通用的

# Polyspace产品的功能

- 第三方IDE环境的集成
  - Visual Studio, Eclipse
- 能够与用户的构建环境进行集成并自动生成Polyspace配置文件
  - polyspace-configure
- 自动代码生成工具的集成
  - Embedded Coder, Target Link, IBM Rational Rhapsody
- 支持在集群上进行部署安装
  - 利用集群优势加快Polyspace Code Prover的验证速度

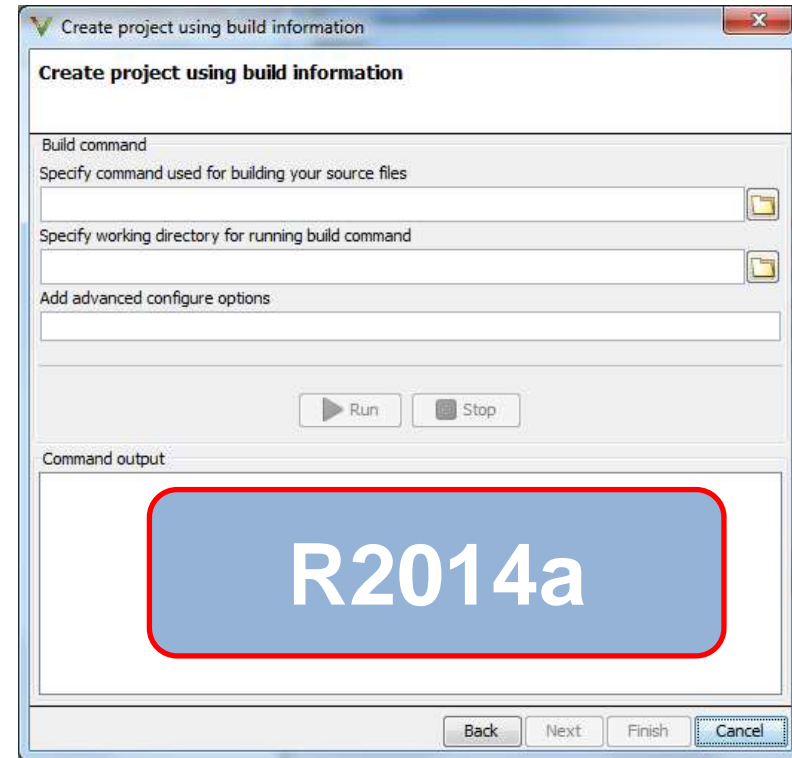
# Polyspace支持与第三方IDE环境集成



- 支持的環境有Eclipse, Visual Studio

# Polyspace配置工具：polyspace-configure

- 与用户的构建环境进行集成，自动生成Polyspace工程或配置文件
- 命令：polyspace-configure
- 使用方法：

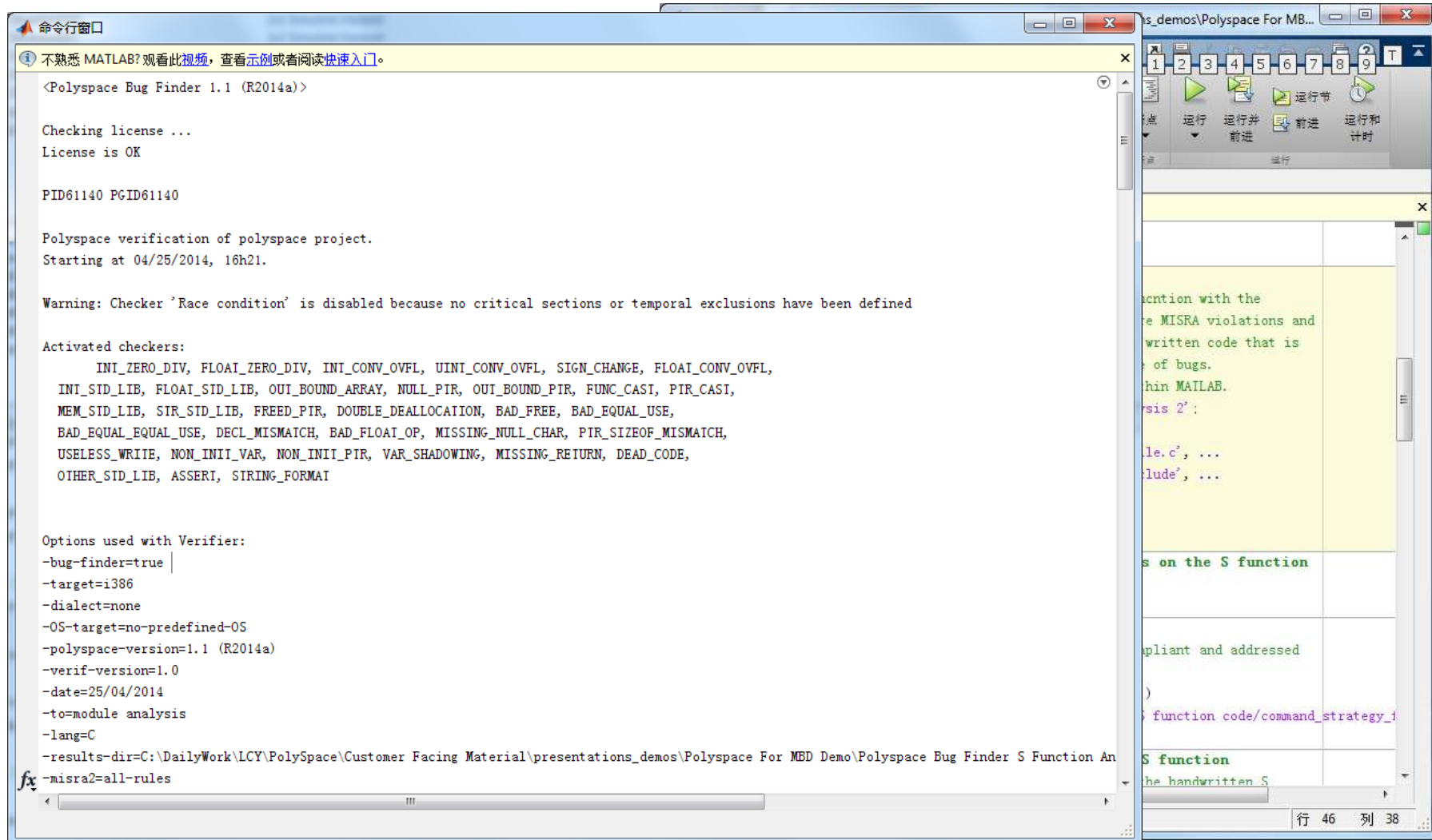


```
polyspace-configure -prog myProject make -B targetName buildOptions
```

[Video](#)

# Polyspace支持与自动代码生成工具进行集成

- Embedded Coder, TargetLink, IBM Rational Rhapsody



The image shows two overlapping windows from the Polyspace software. The foreground window is the '命令窗口' (Command Window) for Polyspace Bug Finder 1.1 (R2014a). It displays the following text:

```

<Polyspace Bug Finder 1.1 (R2014a)>

Checking license ...
License is OK

PID61140 PGID61140

Polyspace verification of polyspace project.
Starting at 04/25/2014, 16h21.

Warning: Checker 'Race condition' is disabled because no critical sections or temporal exclusions have been defined

Activated checkers:
    INI_ZERO_DIV, FLOATI_ZERO_DIV, INI_CONV_OVFL, UINI_CONV_OVFL, SIGN_CHANGE, FLOATI_CONV_OVFL,
    INI_SID_LIB, FLOATI_SID_LIB, OUI_BOUND_ARRAY, NULL_PTR, OUI_BOUND_PTR, FUNC_CASI, PIR_CASI,
    MEM_SID_LIB, SIR_SID_LIB, FREED_PTR, DOUBLE_DEALLOCATION, BAD_FREE, BAD_EQUAL_USE,
    BAD_EQUAL_EQUAL_USE, DECL_MISMATCH, BAD_FLOATI_OP, MISSING_NULL_CHAR, PIR_SIZEOF_MISMATCH,
    USELESS_WRITE, NON_INIT_VAR, NON_INIT_PTR, VAR_SHADOWING, MISSING_RETURN, DEAD_CODE,
    OTHER_SID_LIB, ASSERTI, SIRING_FORMATI

Options used with Verifier:
-bug-finder=true |
-target=i386
-dialect=none
-OS-target=no-predefined-OS
-polyspace-version=1.1 (R2014a)
-verif-version=1.0
-date=25/04/2014
-to=module analysis
-lang=C
-misra2=all-rules
  
```

The background window is a code editor showing a snippet of C code with green annotations. The visible code includes:

```

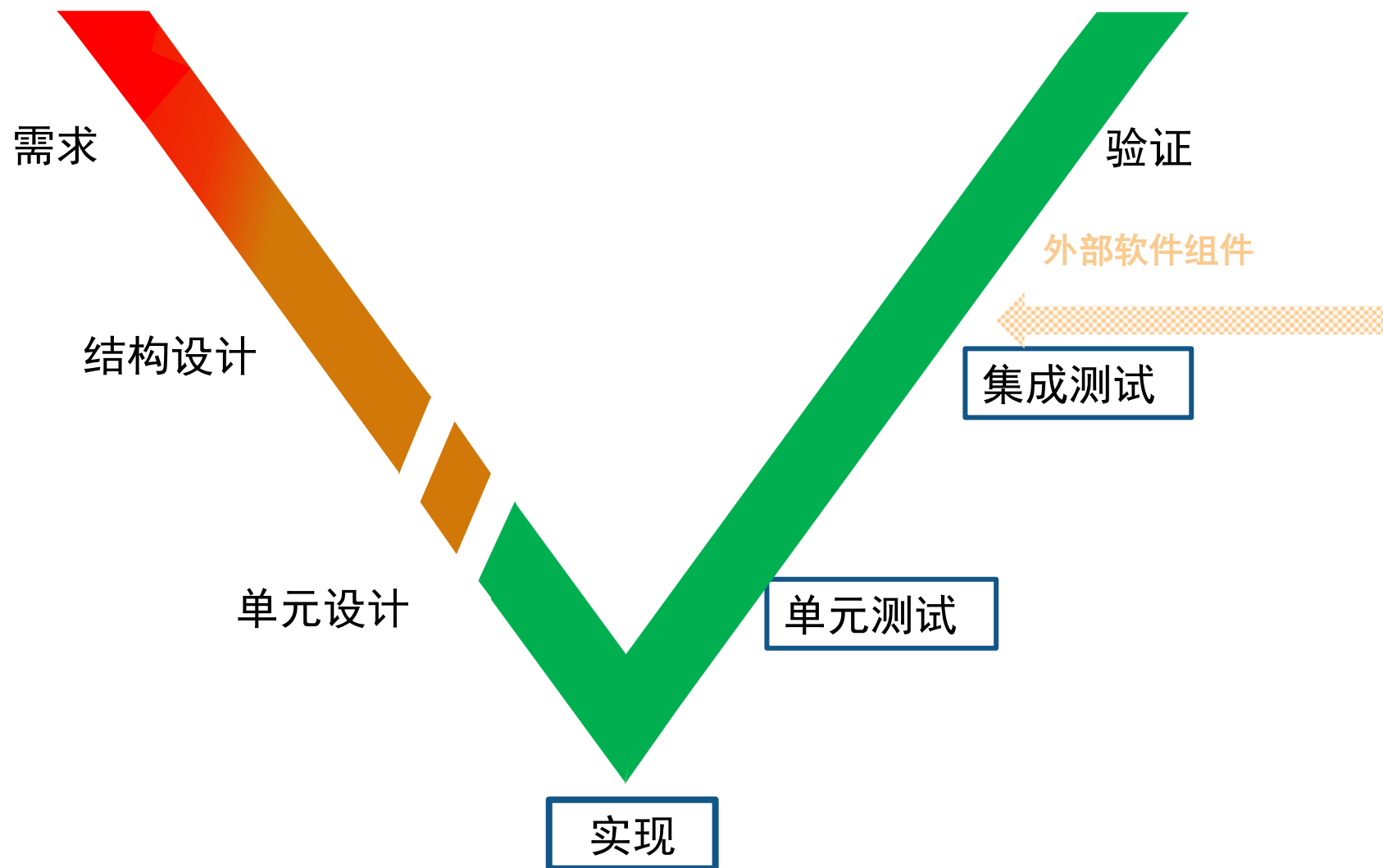
...
function with the
...
the MISRA violations and
...
written code that is
...
of bugs.
...
thin MATLAB.
...
sis 2':
...
le.c', ...
...
lude', ...
...
s on the S function
...
pliant and addressed
...
)
...
function code/command_strategy_f
...
S function
...
the handwritten S
  
```

The code editor window has a toolbar with buttons for '运行' (Run), '运行并前进' (Run and Advance), '运行并计时' (Run and Time), and '运行并前进计时' (Run and Advance Time). The status bar at the bottom right of the code editor shows '行 46 列 38' (Line 46, Column 38).

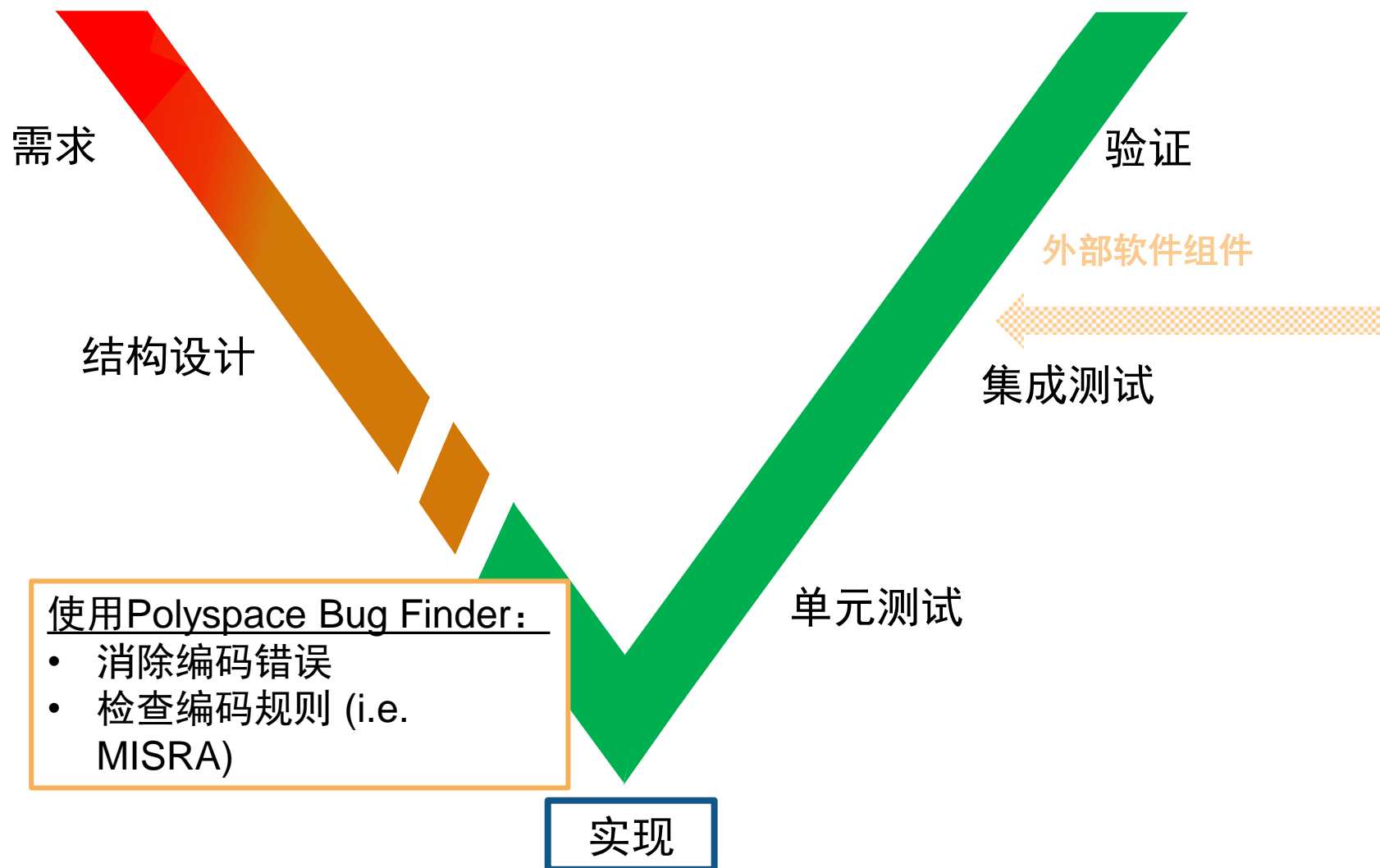
# 内容

- Polyspace背景介绍
- Polyspace产品功能及使用技巧
- 在基于模型设计开发流程中使用Polyspace
- Polyspace及相关工具对认证的支持

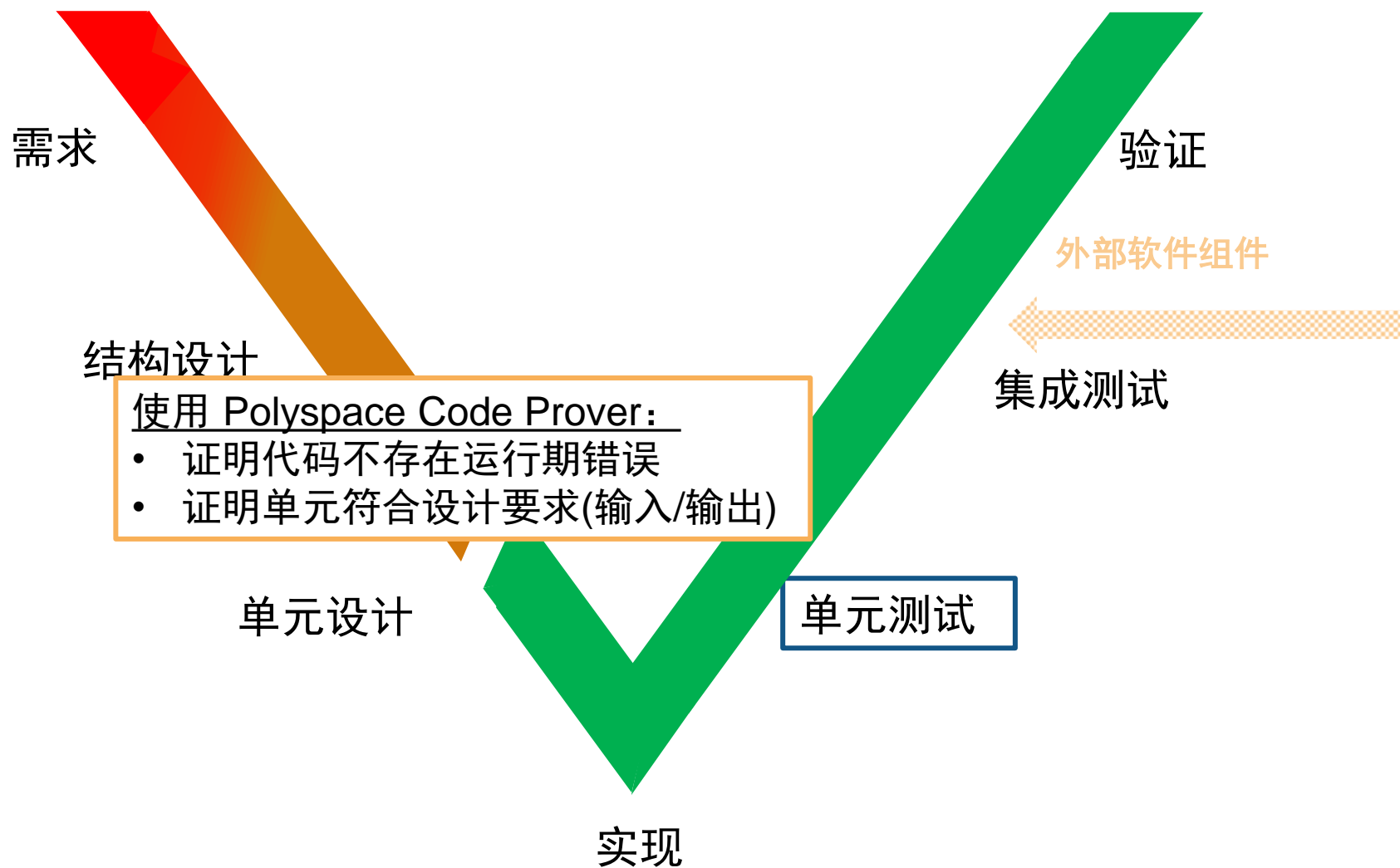
# 在开发流程中引入Polyspace



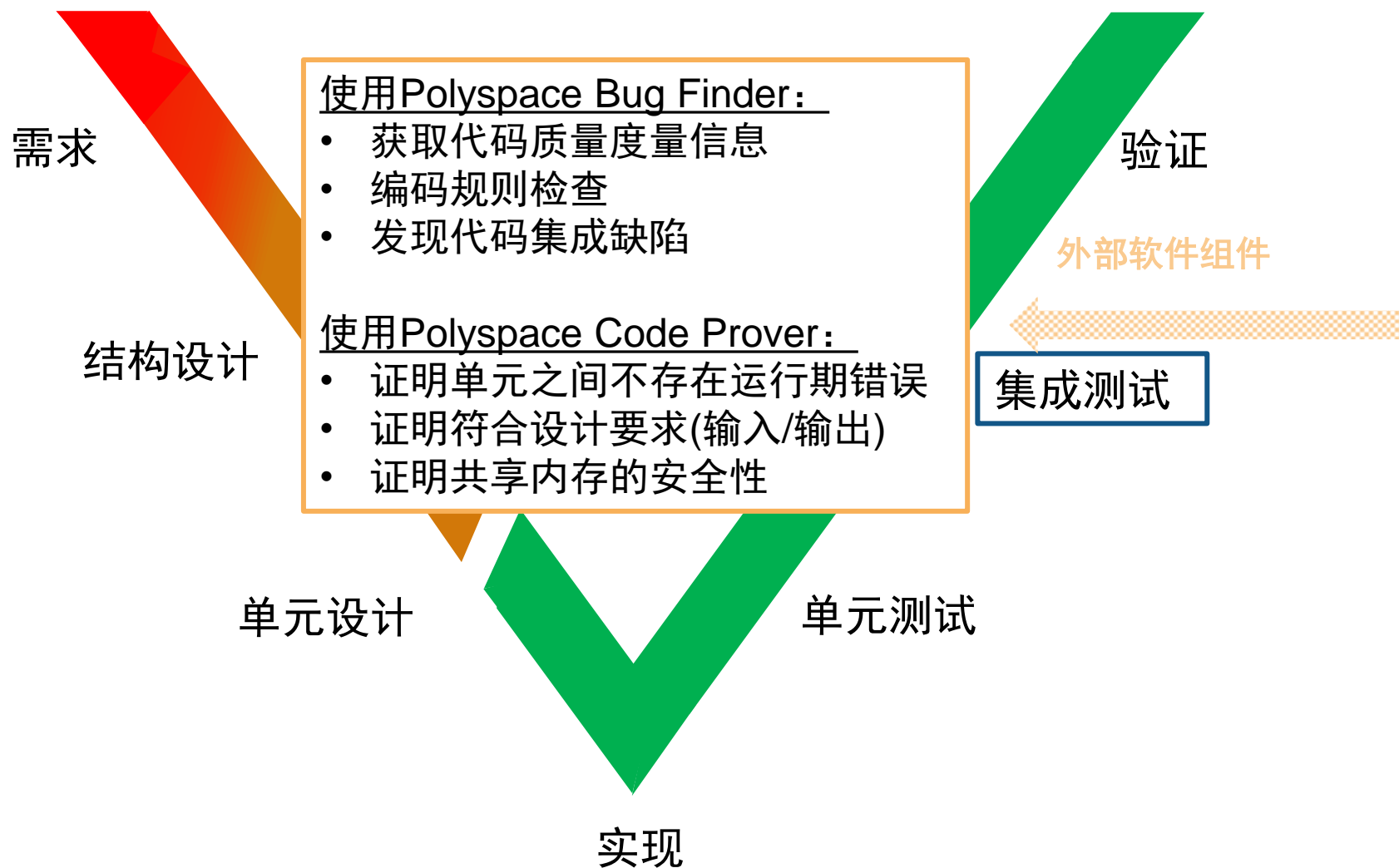
# 在开发流程中引入Polyspace



# 在开发流程中引入Polyspace



# 在开发流程中引入Polyspace



# 在开发流程中引入Polyspace

需求

## 发布Polyspace 结果

- Polyspace Web Metrics
- 生成报告



单元设计

Table of Contents	
1. Polyspace Code Verification Summary	1
Project Information	1
2. Verification Characteristics	2
Verification Characteristics for: Demo_C_Workflow - Result_version_09_DataRange_Added_1	2
3. Code Metrics Summary	4
Code Metrics Summary for: Demo_C_Workflow - Result_version_09_DataRange_Added_1	4
4. Coding Rules Summary	6
MISRA-C Coding Rules SQO Summary for: Demo_C_Workflow - Result_version_09_DataRange_Added_1	6
5. Polyspace Run-Time Checks	8
Run-Time Checks SQO Summary for: Demo_C_Workflow - Result_version_09_DataRange_Added_1	8
Systematic Run-Time Violations	8
Non-Terminating Function Calls and Loops	8
Unreachable Branches	8
Potential Run-Time Violations	8
6. Appendix 1 - Configuration Settings	16
Polyspace Settings	16
DRS Configuration Data	12
DRS - Global Variables	12
DRS - Standard Functions	12
Coding Rules Configuration	12
Software Quality Objectives Configuration	17
Code Metrics	17
Coding Rules	18
Run-Time Checks	20
7. Appendix 2 - Code Metrics Details	22
Code Metrics Details for: Demo_C_Workflow - Result_version_09_DataRange_Added_1	22
Project Metrics for: Demo_C_Workflow - Result_version_09_DataRange_Added_1	23
File Metrics for: Demo_C_Workflow - Result_version_09_DataRange_Added_1	23
Function Metrics for: Demo_C_Workflow - Result_version_09_DataRange_Added_1	23
8. Appendix 3 - MISRA-C Coding Rules Details	29
MISRA-C Summary for all Files	29
MISRA-C Warnings	30
9. Appendix 4 - Run-Time Check Details	36
Systematic Run-Time Violations	36
Non-Terminating Function Calls and Loops	36
Unreachable Branches	36

**Polyspace Code Verification**  
**SQO Report for: Demo\_C\_Workflow**  
 Report Author: cpreve

# 内容

- Polyspace背景介绍
- Polyspace产品功能及使用技巧
- 在基于模型设计开发流程中使用Polyspace
- Polyspace及相关工具对认证的支持

# 软件质量目标(Software Quality Objectives)

- 在Polyspace中设定软件质量目标
  - 确定文件、模块或者组件是否达到了预定质量目标
- 定义软件质量目标门槛
  - 软件代码度量信息
  - 违反编码规则数量
  - 红色、灰色、橙色的检查数量
- 将SQO作为流程指南
  - 根据质量目标来指定工具使用和流程改进的计划

Software Quality Objectives					
Overall Status	Level	Review Progress	Code Metrics over Threshold	Justified Coding Rules	Justified Run-Time Errors
FAIL	SQO-6	81.0%	2	100.0%	50.0%
FAIL	6 finding(s) over threshold		2	100.0%	100.0%

# 使用Polyspace 进行DO-178 认证

MathWorks can provide a DO-178 Qualification Kit to help qualify Polyspace for your DO-178 certification projects



- 使用DO Qualification Kit, 您可以
  - 证明Polyspace 产品可以用于满足DO-178C 和相关标准如DO-330产品开发活动
  - 帮助获得认证
- 套件包括：
  - 文档，测试用例和流程指导书
  - 工具资质证明计划，工具的使用要求和其它材料
  - 允许针对特定项目对工具认证套件进行修改

# Polyspace支持DO-178B/C认证

- 代码与软件设计结构一致 (architecture)
  - 数据字典, 函数调用树
- 代码是可以被验证的 (Verifiable)
  - 发现不可达代码
- 代码要符合编码规则
  - MISRA C, MISRA C++, 或 JSF++ 规则检查
  - 静态度量信息, 例如圈复杂度 (编码标准的一部分)
- 代码是准确的 (Accurate)和一致的 (Consistent)
  - 发现代码中的运行期错误, 例如除零、溢出、变量未初始化等

# 使用Polyspace for IEC 61508, ISO 26262, EN 50128, IEC 62304认证

MathWorks can provide a Certification Kit for IEC 61508 related standards



- 使用认证套件, 您可以
  - 在推荐流程下使用Polyspace来验证您的代码
  - 帮助获得认证
  
- 套件包括
  - 工具资质证书, 审核材料和测试套件
  - 可以使用Polyspace的工具证明材料
  - 推荐的工作流程



# Polyspace可以提供的帮助

