AdaCore
The GNAT Pro Company

GNAT Pro Innovations for High-Integrity Development

José F. Ruiz <ruiz@adacore.com>
Senior Software Engineer

Ada Europe 2010, Valencia

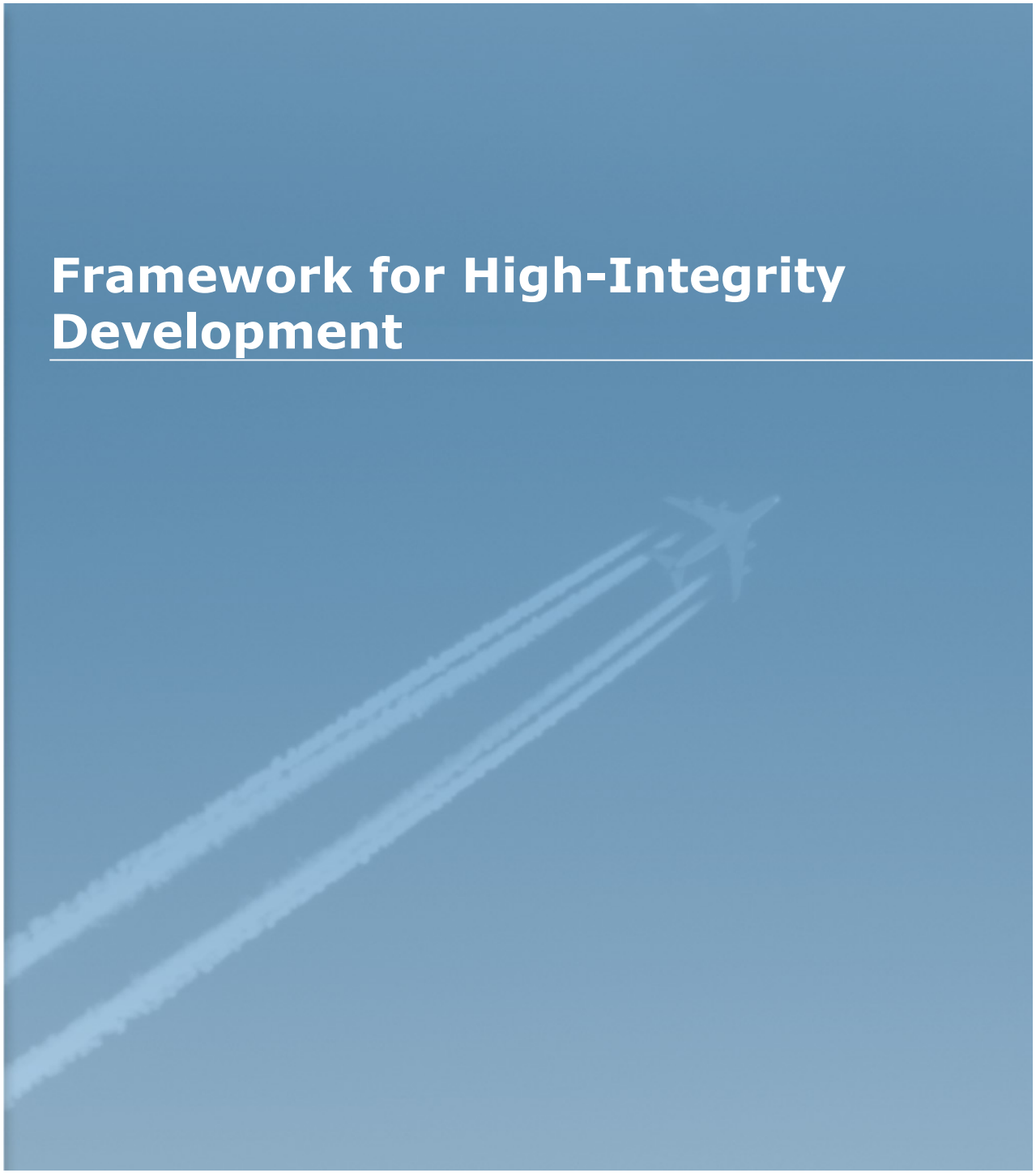
2010-06-15

Index

- **Development environment**
- **Tools**
  - Static analysis
  - Dynamic analysis
- **Research projects**
- **Open initiatives**

AdaCore
The GNAT Pro Company

Framework for High-Integrity Development



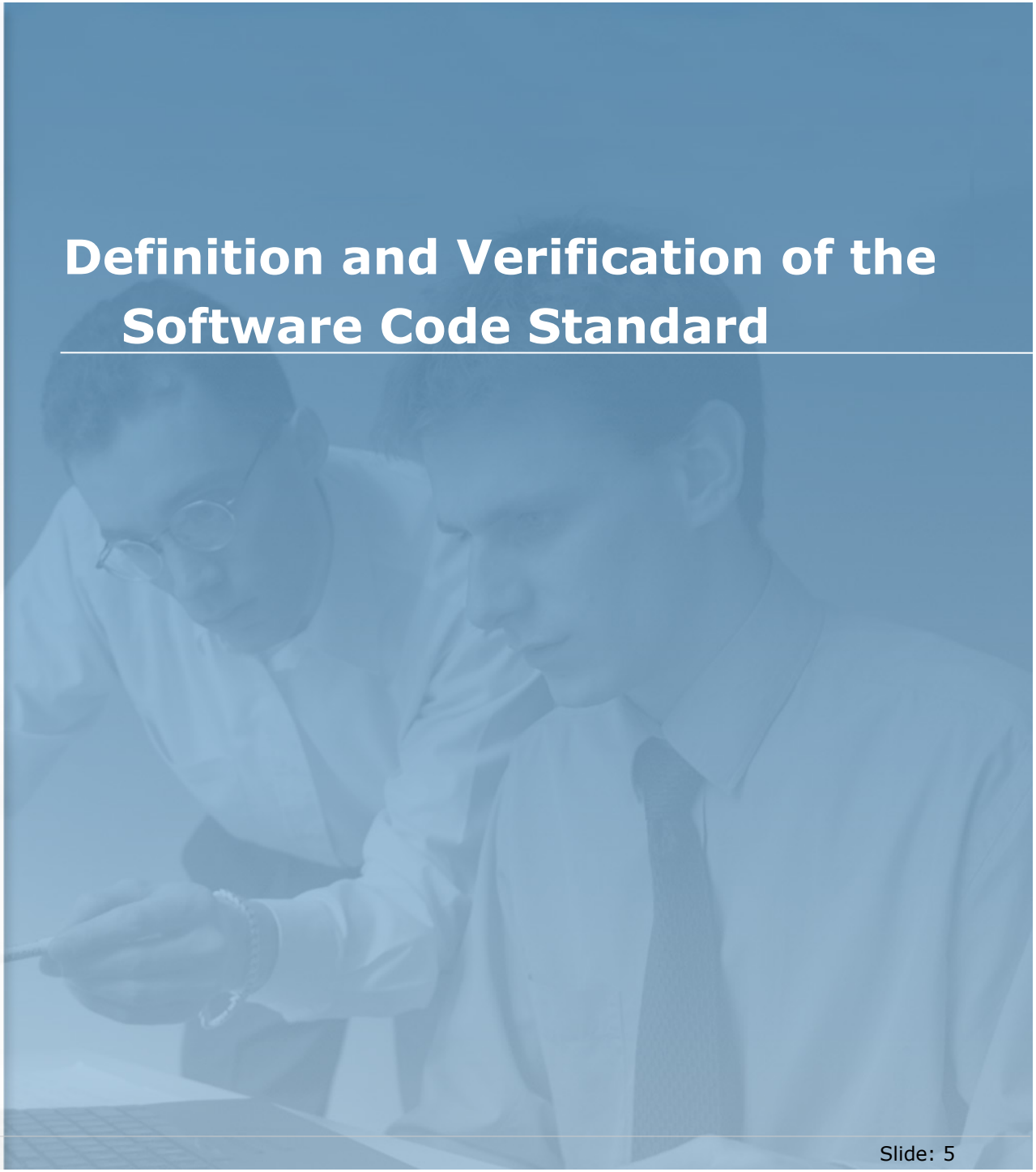
GNAT Pro High-Integrity Edition Package Overview

- **For High-Integrity Development**
  - Reduced-Footprint & Certifiable Ada Run-Times
  - Ada Run-Time Customization Capability
  - Support & On-Line Consulting Adapted to the Requirements of Safety-Critical Application Development
  - Certified/qualified run times
    - Certification Material (DO-178B, Level A) for the Cert Run-Time for Wind River PSC ARINC653
    - Qualification Material (ECSS-E-ST-40C and ECSS-Q-ST-80C, Level B) for the Ravenscar Run-Time for ERC32, LEON2, LEON3 (in progress)

- **Tools for High-Integrity**
  - Coding Standard Analysis
  - Static Stack Analysis
  - Peer Review
  - Design by Contract
  - Code Coverage Analysis
  - Traceability Analysis Kit

AdaCore
The GNAT Pro Company

Definition and Verification of the Software Code Standard



Code Standard: What the DO-178B Standard Says

4.5: SOFTWARE DEVELOPMENT STANDARDS

…

NOTE: … Constraints and rules on development, design and coding methods can be included to **control complexity**. **Defensive programming** practices may be considered to improve robustness.

11.8 SOFTWARE CODE STANDARDS

…

These standards should include:

a. Programming language(s) to be used and/or **defined subset(s)**…

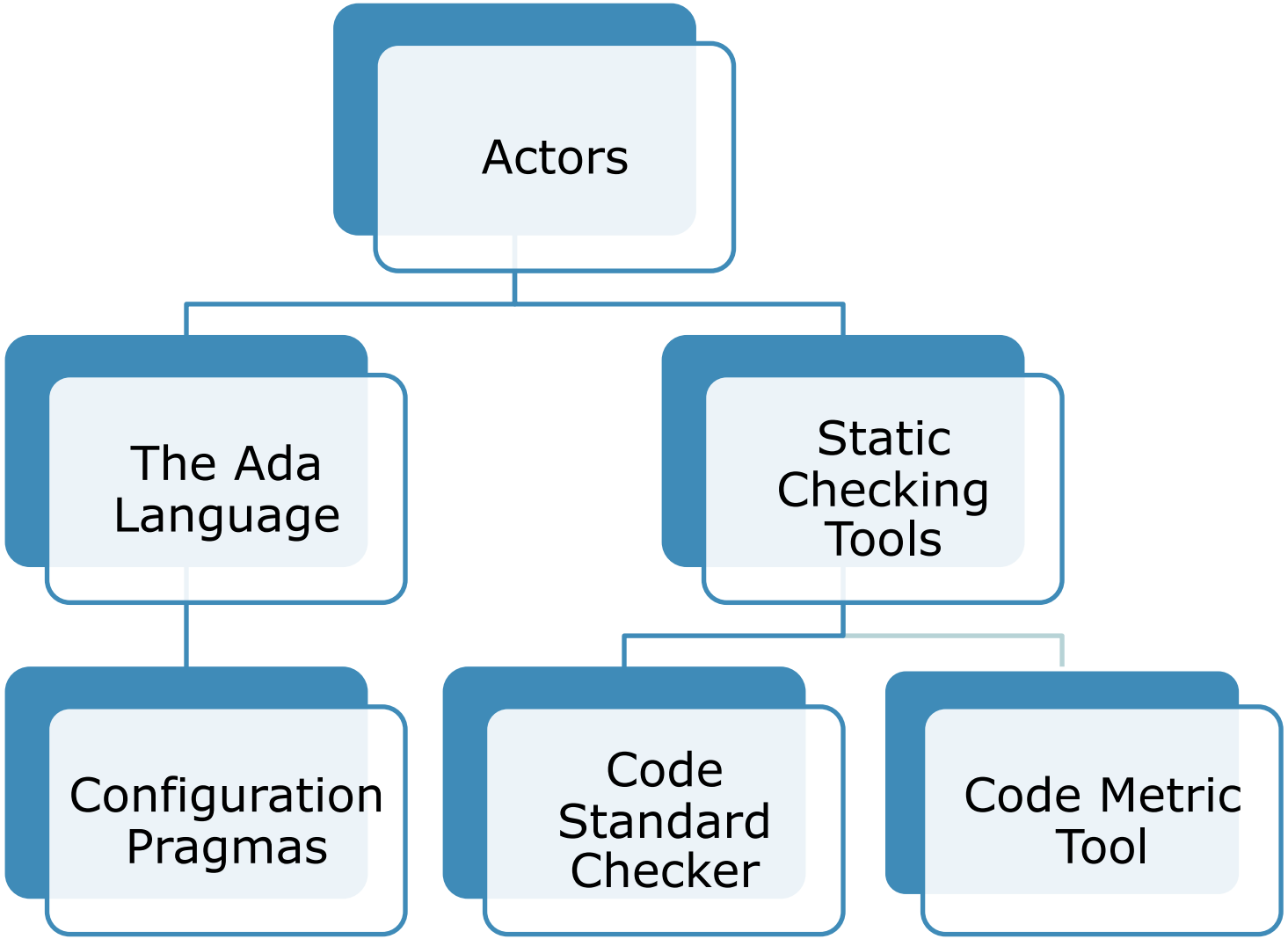
b**. Source Code presentation** standards…

c. **Naming conventions** for … subprograms, variables, and constants.

d. Conditions and constraints imposed on permitted coding conventions…

e. Constraints on the use of the coding tools.

Getting Maximum Leverage from GNAT Pro



Definition of a Code Standard (references to RTCA/DO-178B)

- **Compliance to a Source Code Presentation (11.8.b)**
  - Line length
  - Indentation
- **Controlling complexity (6.3.4.d)**
  - Limit nesting level of control structures
  - Measurement of expression complexity (e.g. McCabe)
- **Ease Control Flow Analysis (6.3.3.b – Reviews and Analyses of the Software Architecture)**
  - Forbid (see also ISO/IEC TR 15942 – 2.3.1):
    - goto statements
    - improper returns
    - exceptions as control flows
    - exit statements exiting from an outer loop
    - Recursions
- **Consistent Naming Conventions**
  - Suffix rules (e.g. _T for type, _C for constants, etc.)

Checking Code Standard

- **At the Tool Level**
  - **GNATcheck**

```
-Rall
+RStyle_Checks: 3                                    -- indentation of 3
    characters
+RStyle_Checks: M79                                  -- line length <= 79
    characters
+ROverly_Nested_Control_Structure:3                  -- nesting level <= 3
+RNo_Goto_Statements
+RImproper_Returns
+RMisnamed_Identifiers: Type_Suffix=_T, Constant_Suffix=_C
+RExceptions_As_Control_Flow
+RGotten_Loop_Exits
```

GNATcheck being **qualified** (DO-178B) as a **verification tool**

Code Metrics

- **Tool support**
  - **GNATmetric**

```
Line metrics summed over 90 units
  code lines          : 4186

Element metrics summed over 90 units
  all statements      : 737
  all declarations    : 2036
  logical SLOC        : 2773

134 public types in 30 units including
    16 private types

166 type declarations in 38 units

246 public subprograms in 41 units

181 subprogram bodies in 32 units

Average cyclomatic complexity: 1.98
```

AdaCore
The GNAT Pro Company

Static Stack Analysis

Why stack analysis?

6.3.4 Reviews and Analyses of the Source Code

[…]

f. Accuracy and consistency: The objective is to determine the correctness and consistency of the Source Code, including **stack usage**, fixed point arithmetic overflow and resolution, resource contention, worst-case execution timing, exception handling, use of uninitialized variables or constants, unused variables or constants, and data corruption due to task or interrupt conflicts.

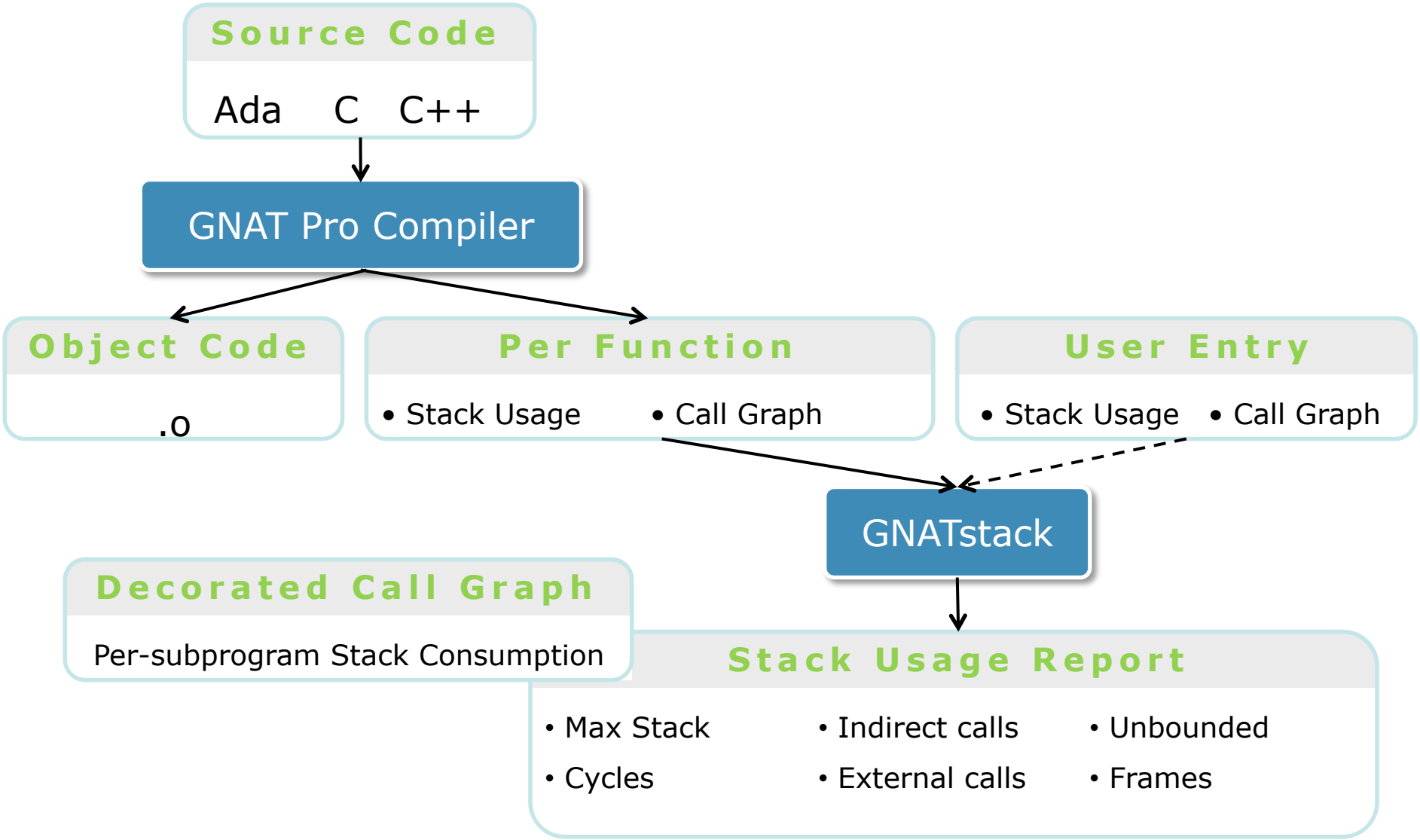
The Static Approach - GNATstack

- **What it brings**
  - Gives an upper Worst Case bound
  - Qualifiable Tool
  - A safe answer to the DO-178B requirement

- **To be aware of**
  - Worst Case may be pessimistic – worst case on a path may never be taken
  - Coding rules (typical safety-critical standards will forbid them)
    - No cycles in the call graph
    - No recursions
    - No unbounded frame
  - User input may be needed
    - Indirect calls
    - External calls

⇒ **Simplifies & automates worst-case stack usage analysis**

GNATstack Workflow

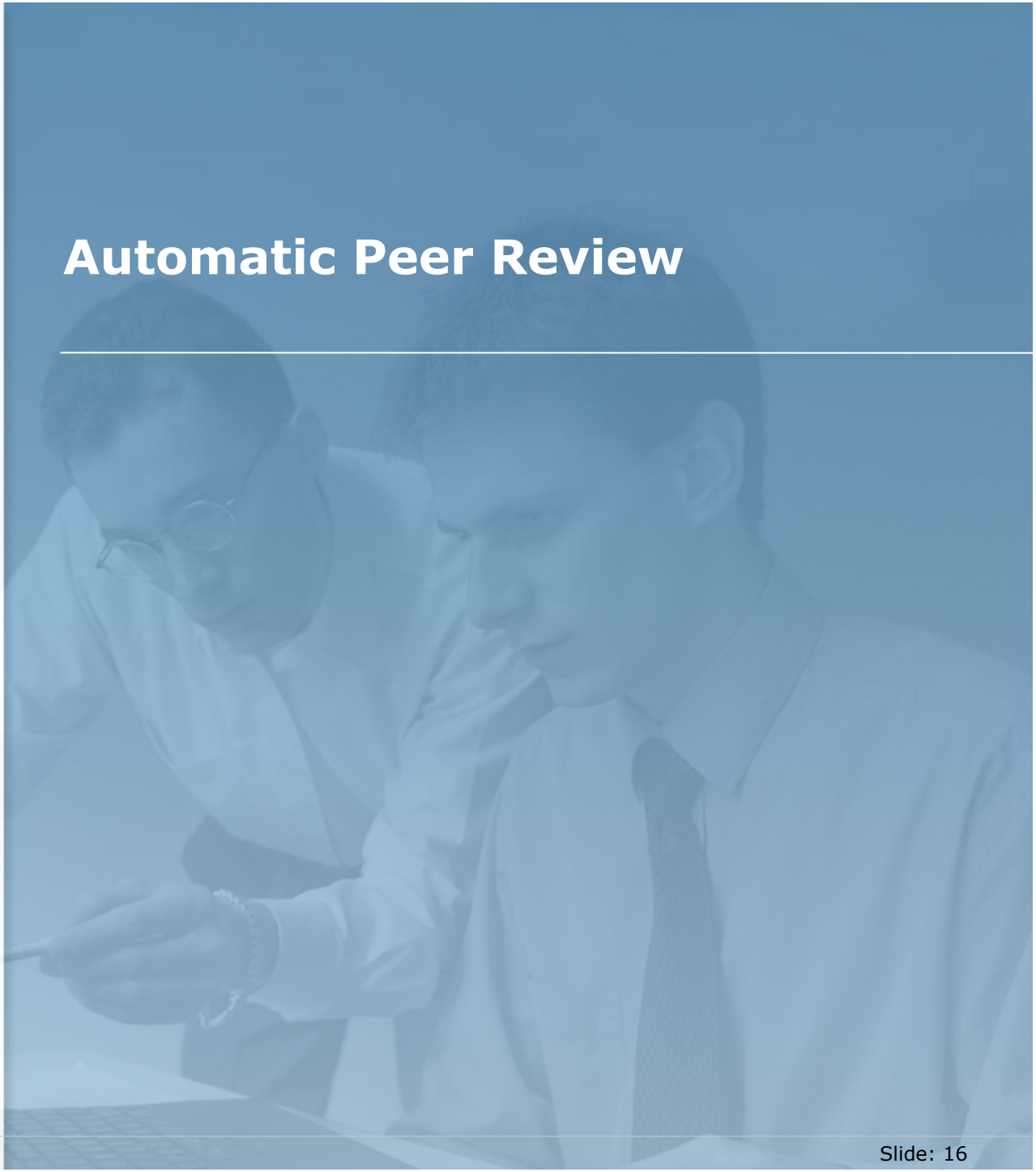


Evolutions

- **Support for Ada, C, C++**
- **Dispatching calls**
  - Extend the compiler to tell
    - Class hierarchies
    - Overloaded primitive operations
    - Root class/method at each dispatching call
  - Global analysis to select the set of possible target calls
  - May be refined manually or by tool doing data-flow analysis
- **Indirect calls**
  - The candidate subprograms for indirect calls are those
    - With compliant profile
    - For which a reference has been taken

AdaCore
The GNAT Pro Company

Automatic Peer Review



Peer Review

6.3.4 Reviews and Analyses of the Source Code

[…]

f. Accuracy and consistency: The objective is to determine the correctness and consistency of the Source Code, including stack usage, **fixed point arithmetic overflow and resolution**, resource contention, worst-case execution timing, **exception handling**, use of uninitialized variables or constants, unused variables or constants, and **data corruption due to task or interrupt conflicts**.

CodePeer in Action

CodePeer

- Static run-time errors detection
- Test vectors generation
- Pre/post conditions generation
- Analysis results consolidation

Day-to-day development

- compile-time analysis
- local analysis

Software maintenance

- global change impact analysis

Project quality assurance

- global analysis
- test vectors leverage

CodePeer Area of Action

Ada Run-Time Checks

- out-of-bound indexing
- numeric overflow
- division by zero
- incorrect invariant

User Checks

- Assert statements
- if … then … raise … control flow

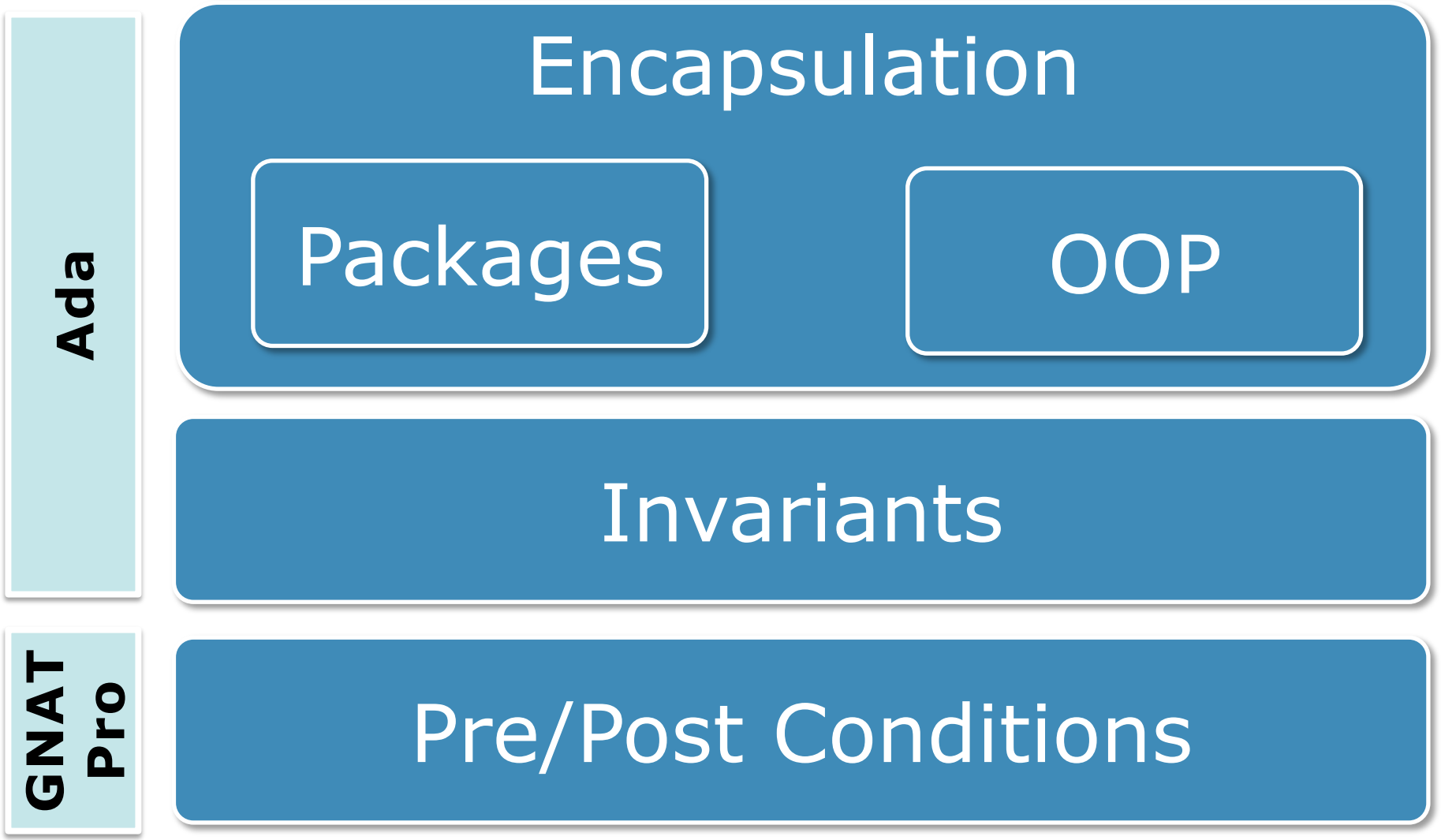
Programming Errors

- Uninitialized variables
- Never ending subps/loops
- Race conditions
- Dead code

AdaCore
The GNAT Pro Company

Design by Contract

Design by Contract



Assertions

- **Part of Ada 2005**

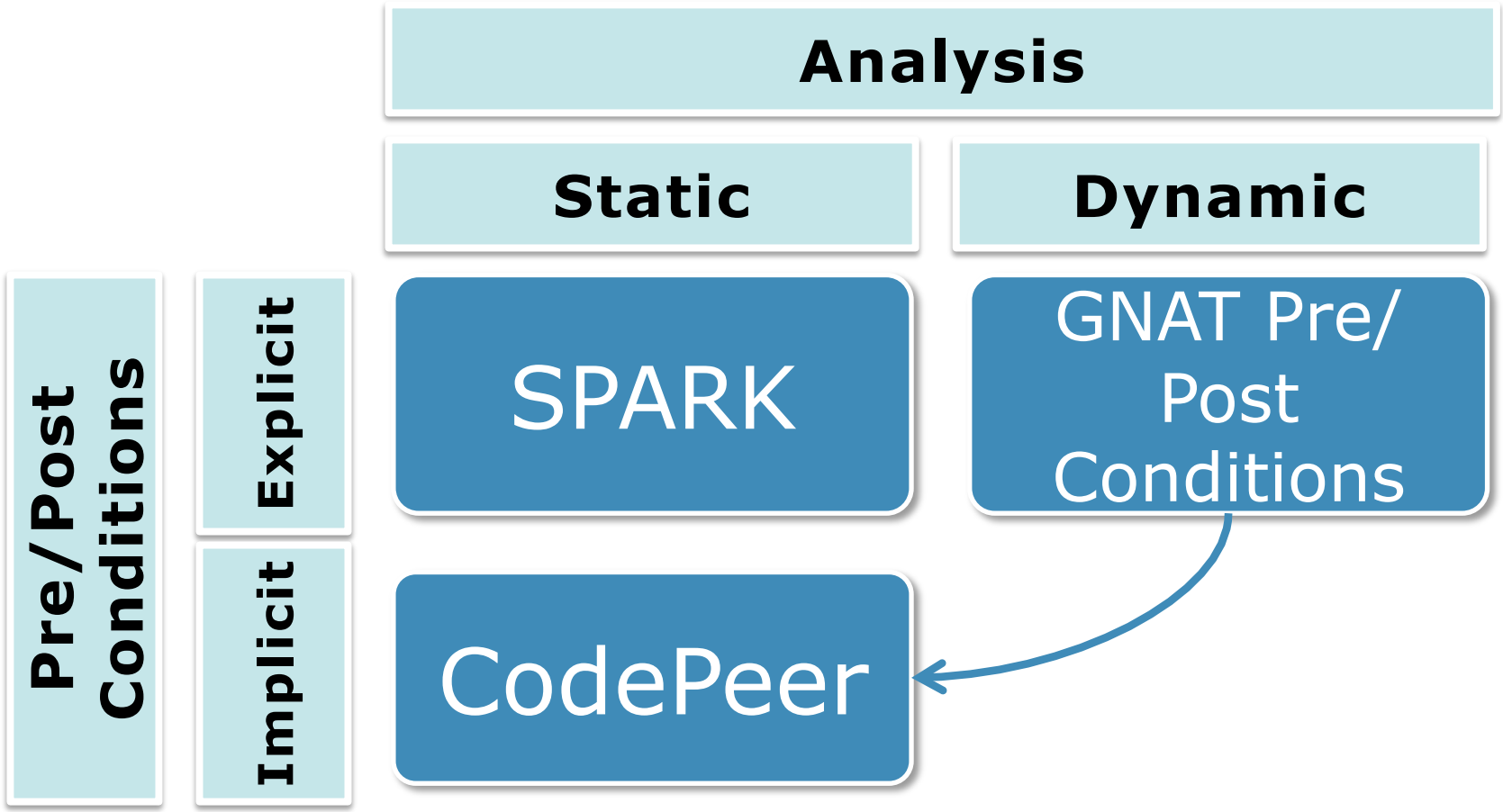
```
procedure Update_Speed_Counter  is
    Speed : Speed_T; -- km/h
begin
    Speed := Get_Speed;
    pragma Assert (Speed <= 280); -- Max theoretical speed  = 261 km/h
end Update_Speed_Counter;
```

- **Activated During Verifications/Tests**
  => Robustness testing
- **Configuration Pragma Assert_Policy to enable/disable assertions**

Pre/Post Conditions with GNAT Pro

- **Tightens the Contract between Caller and Callee**
- **Additional Semantic Information**
  - Can be used for extra checks
- **Checks that the Contract is Respected**
  - At run time
  - Statically

Various Uses of Pre/Post Conditions



Pre/Post Conditions with GNAT Pro

```
package Data_Processing is
   type Data_T is private;
   type Data_A is access Data_T;
   type Links_T is record
         Link1 : Data_A;
         Link2 : Data_A;
   end record;
   function Processing (Param : Data_T) return Links_T;
   pragma Precondition (Param.Link1 not null or else Param.Link2 not null);
   pragma Postcondition
         (Processing'Result.Link1 not null or else Processing'Result.Link2 not
   null);
end Data_Processing;
```

SPARK

- The Correctness-by-Construction approach

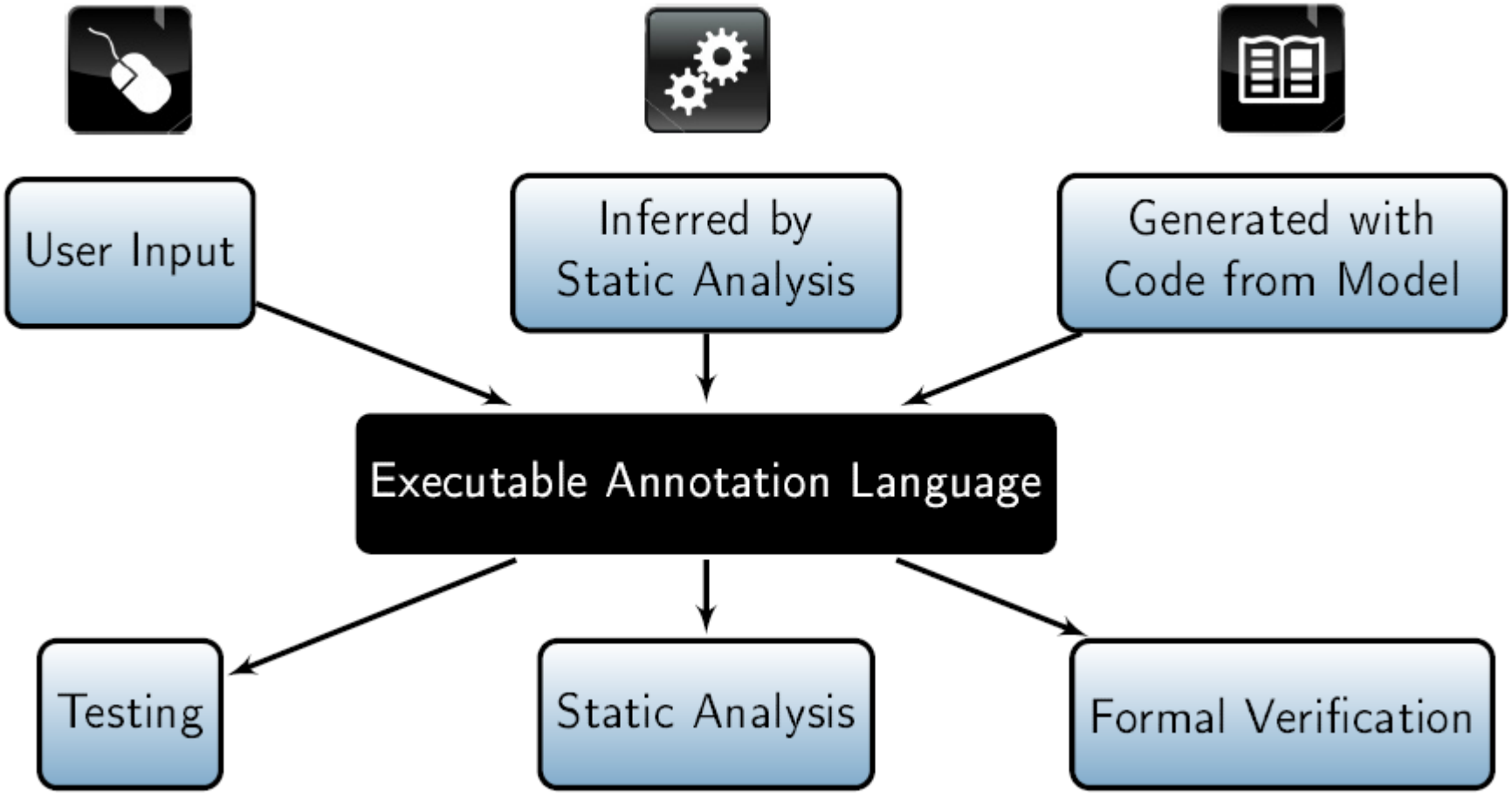
- … more after the coffee

Hi-Lite

- **New research project**
  - AdaCore with Altran Praxis, Astrium Space Transportation, CEA-LIST, the ProVal team of INRIA and Thales Communications
- **Goal**
  - Simplify the use of formal methods
- **How**
  - Loose integration of formal proofs with testing and static analysis
  - Modular
    - Divide-and-conquer approach
  - Early adoption by all programmers in the software life cycle
  - Mixed Ada/C

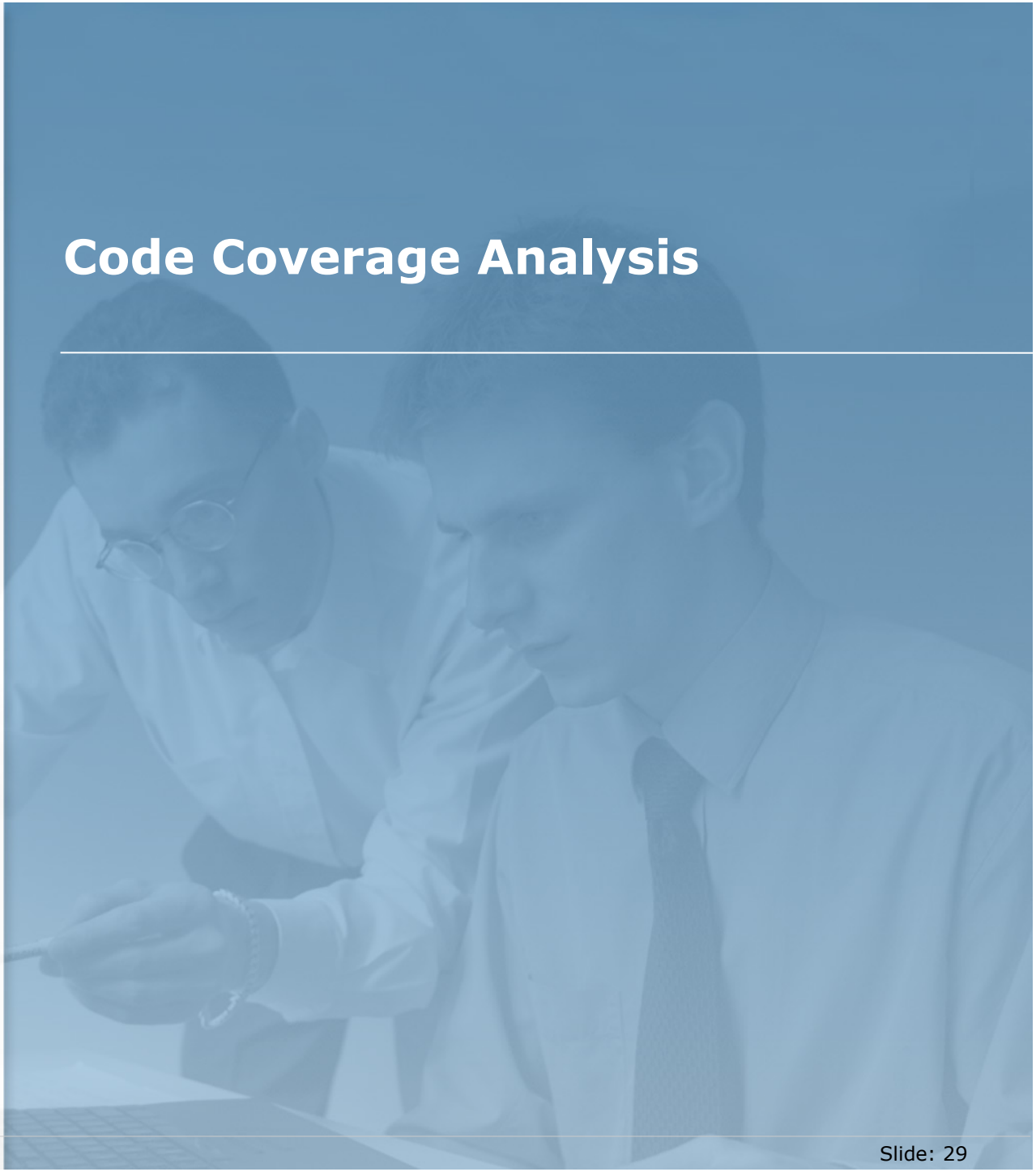
http://www.open-do.org/projects/hi-lite

Hi-Lite: Common Language for Properties



AdaCore
The GNAT Pro Company

Code Coverage Analysis



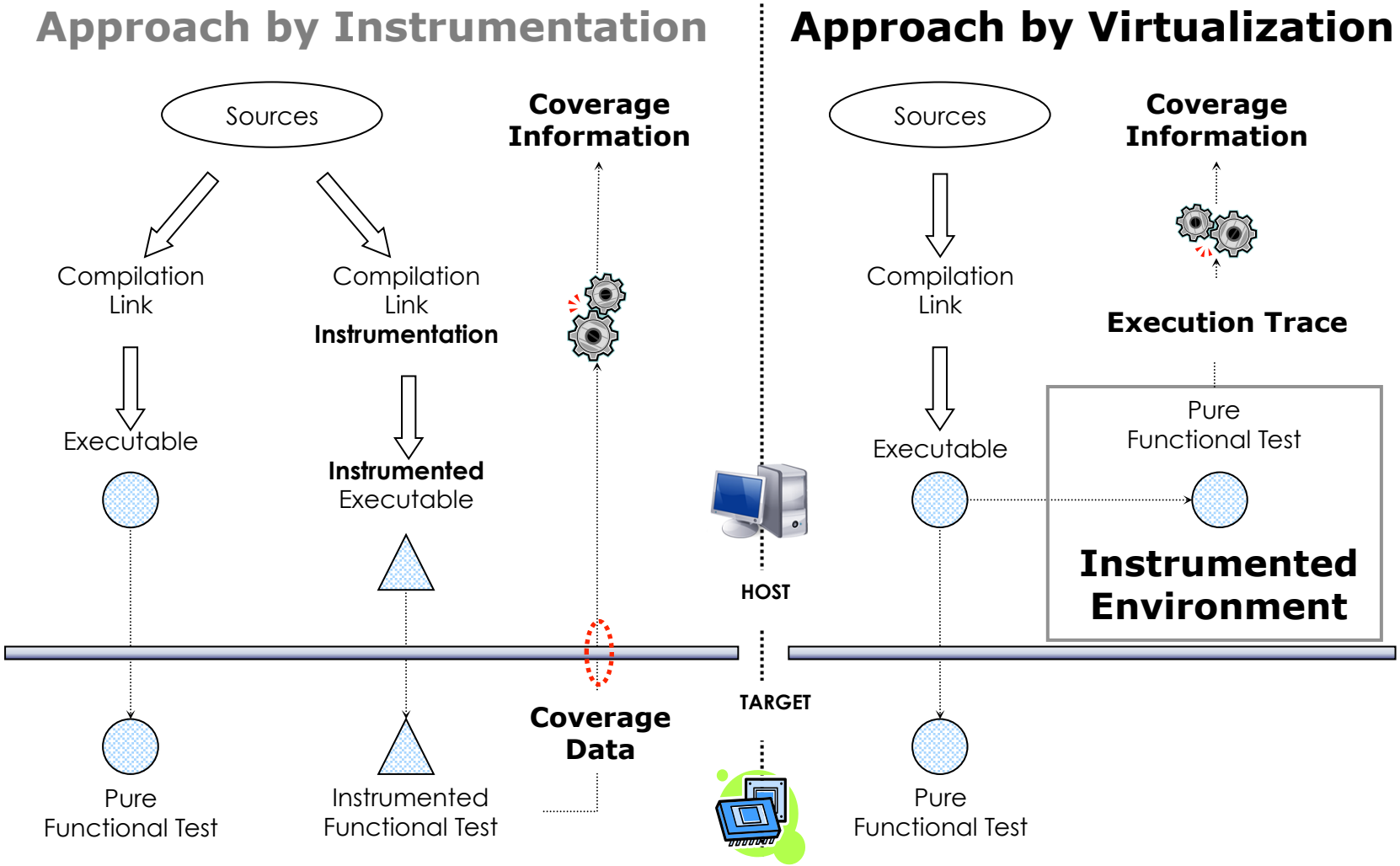
References to the Standard

Table A-7

Verification Of Verification Process Results

	Objective		Applicability by SW Level			
	Description	Ref.	A	B	C	D
	…					
5	Test coverage of software structure (modified condition/ decision) is achieved.	6.4.4.2	●			
6	Test coverage of software structure (decision coverage) is achieved.	6.4.4.2a 6.4.4.2b	●	●		
7	Test coverage of software structure (statement coverage) is achieved.	6.4.4.2c	●	●	○	
	…					

Bridging the Gap between Source and Object Code Coverage



http://www.open-do.org/projects/couverture

Achievements of the Project "Couverture"

Coverage Levels

- **Object:**
  - Instruction
  - Object branch
- **Source**
  - Statement
  - Decision
  - Modified Condition/Decision

Aggregated Coverage

- **Capitalization:** multiple executions
- **Consolidation:** different executables exercising the same function

Tool Qualification

- **For use in DO-178B Level A certification activities**

Emulator

- **Open-Source simulation solution: QEMU**
- **Efficient Simulation Technology**
- **Development platform close to the final HW platform**
  - PowerPC / LEON
- **Code tested can be the code running on the final target**
- **Avoids endianness issues**
- **Eases functional testing**

AdaCore
The GNAT Pro Company

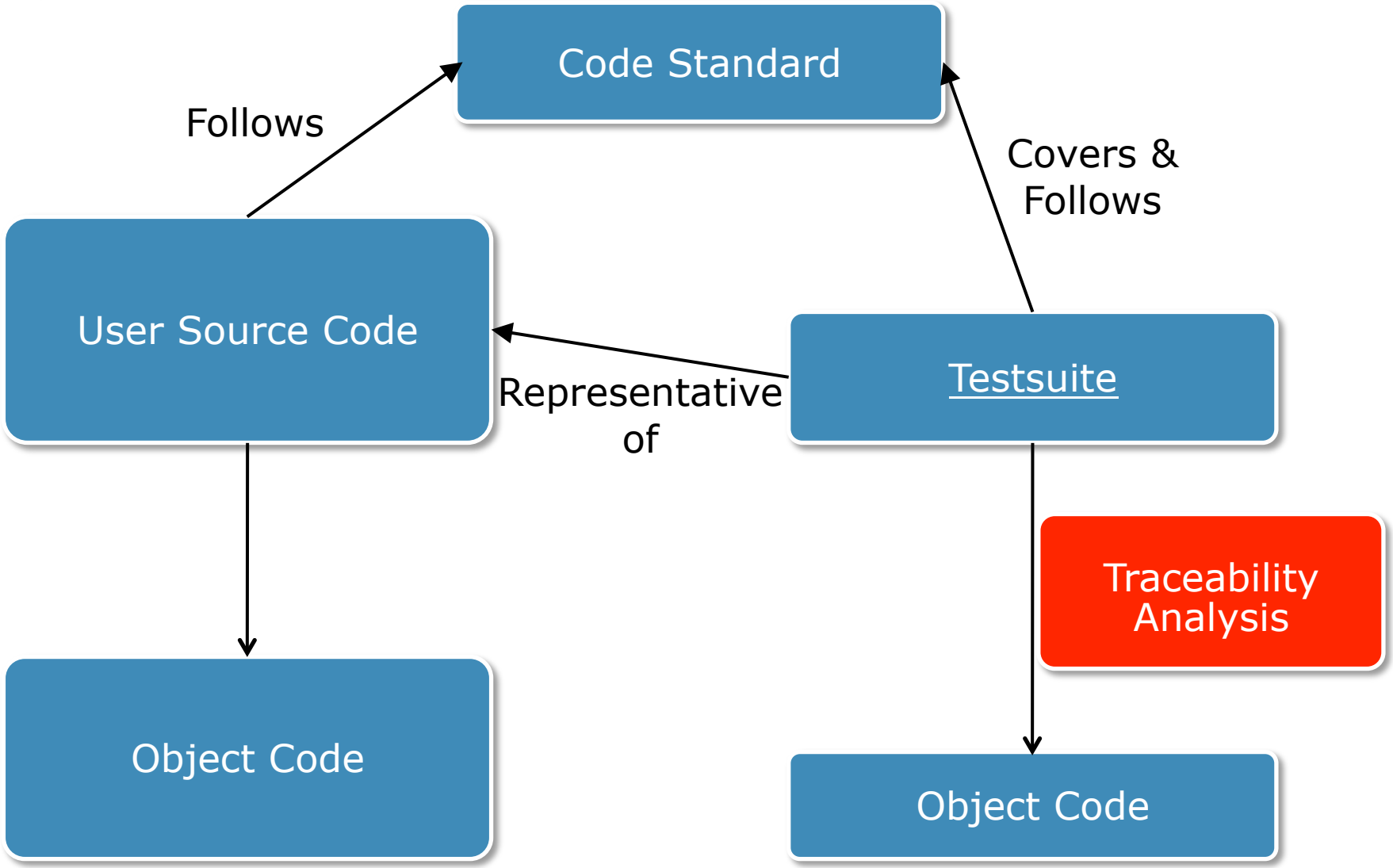
Traceability Analysis Kit

From Source Code to Object Code

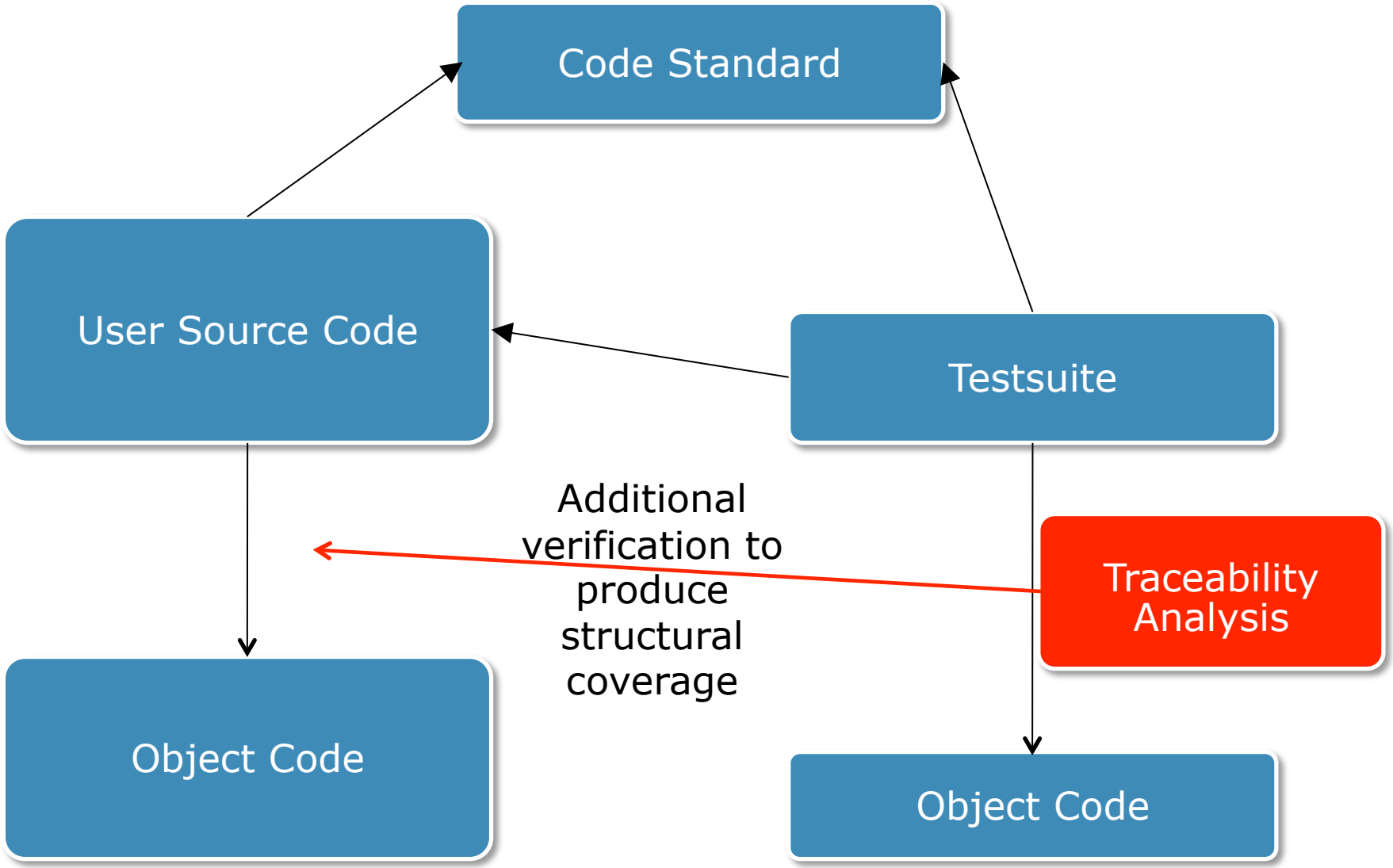
6.4.4.2 Structural Coverage Analysis
[…]
The structural coverage analysis may be performed on the **Source Code**, **unless the software level is A** and the compiler generates object code that is not directly traceable to Source Code statements….

- **Option 1**
  - Code coverage at the source level
  - Traceability analysis on the application source code
- **Option 2**
  - Code Coverage at the object level
  - Bypass of the source to object code traceability analysis
- **Option 3**
  - Code coverage at the source level
  - Traceability analysis on a representative testsuite

Traceability Package



Traceability Package

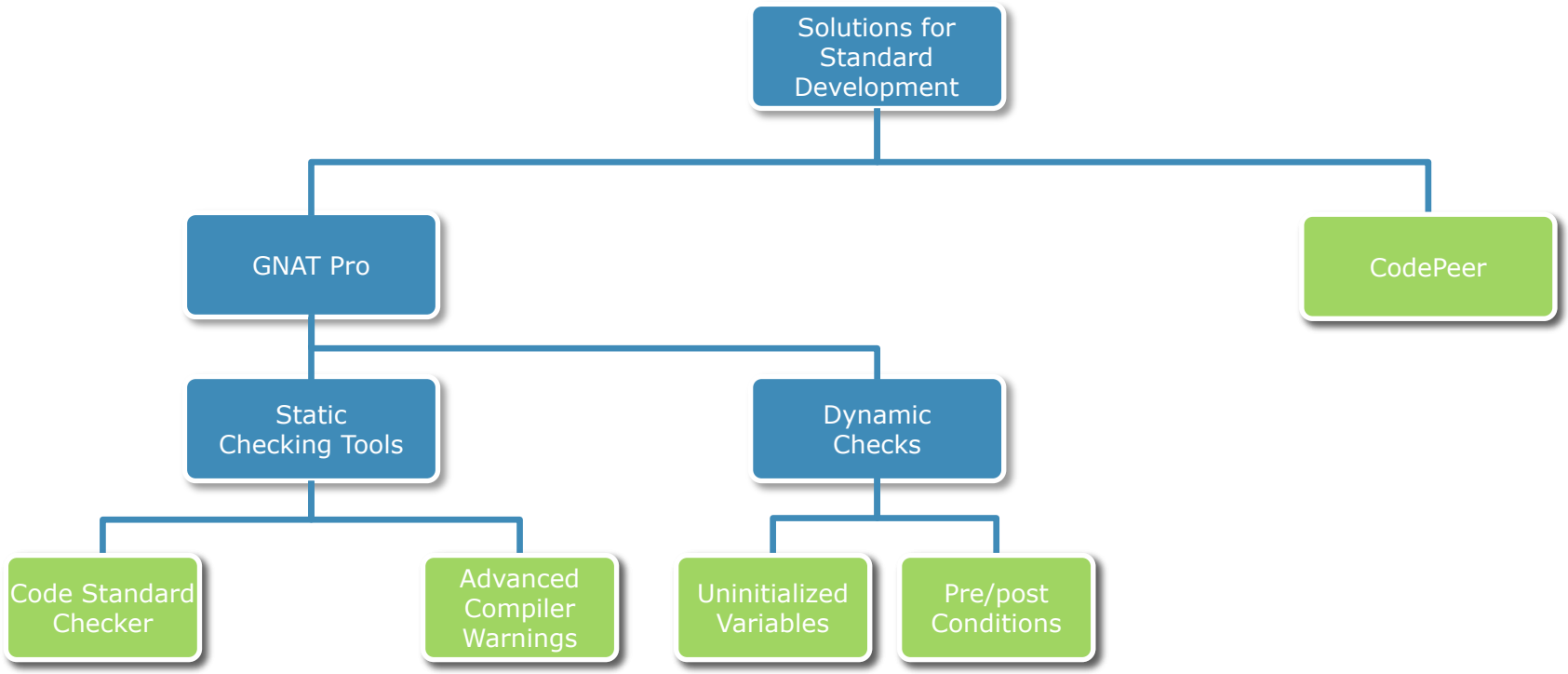


Validity of a Traceability Analysis Package

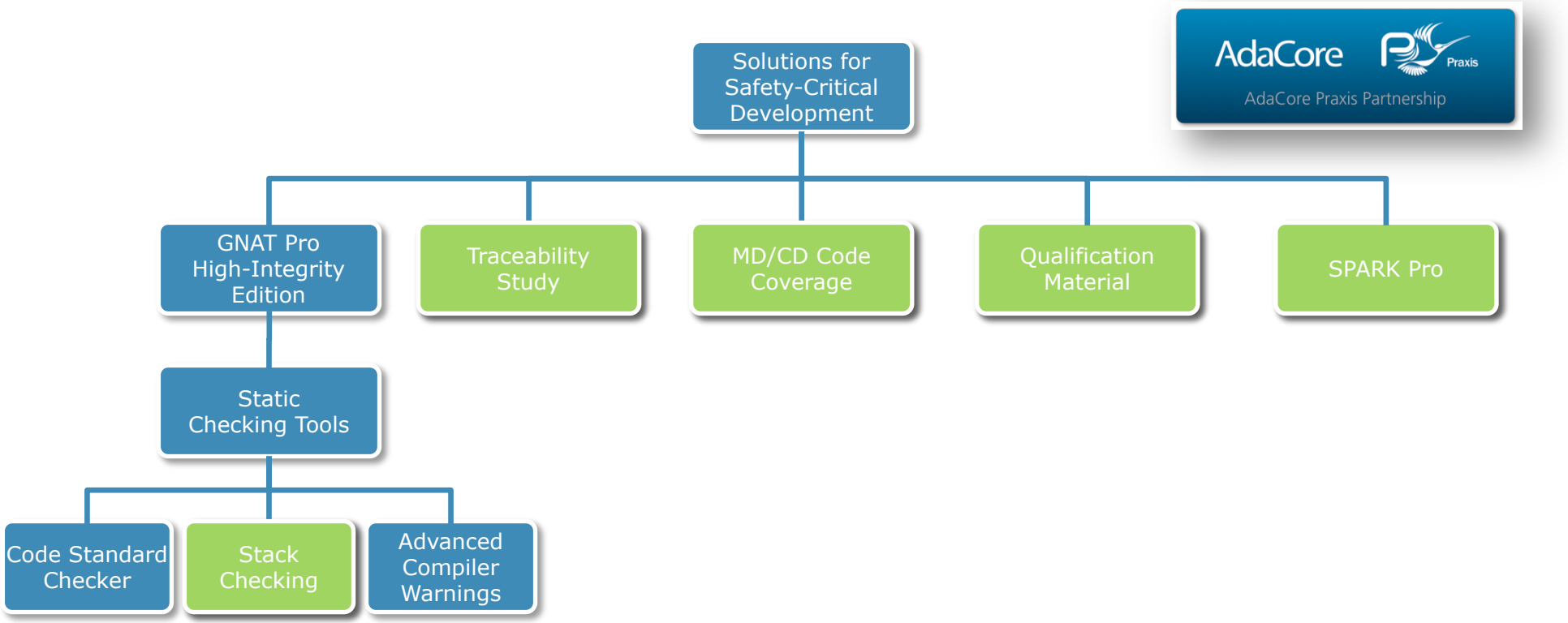
- **The answer comes from the "Final Report for Clarification of DO-178B [...]" (RTCA/DO-248B)**

> **4.12.2.2 Approach to the Traceability Analysis**
> […]
> One approach to this analysis is to produce some **code with fully representative language constructs of the application** (e.g., case statements, if-then-else statements, loops) and to **analyze the resultant object code**. Also, it is important that the individual constructs are completely representative of the constructs used in the target object code build, including complex structures (e.g., nested routines). […] The choice of the representative code constructs should be **agreed with the appropriate certification authority**. […]

Safer Programming: the AdaCore Answer



Safer Programming: the AdaCore Answer



AdaCore
The GNAT Pro Company

Executable Requirements (XReq)

XReq for DO-178B

- **Executable Requirements for Ada, C and C++**
  - Put together the requirements and the tests
  - Tests written in English
  - HLT and LLT automatically generated in Ada, C or C++
- **Help to bring agile to critical projects using Behavior Driven Development (BDD)**
- **Open Sourced by SOGILIS in the Open-DO forge**

Feature: Calculate
In order to do simple calculations
As a user
I want to be able to do simple operations (add, subtract, multiply and divide)

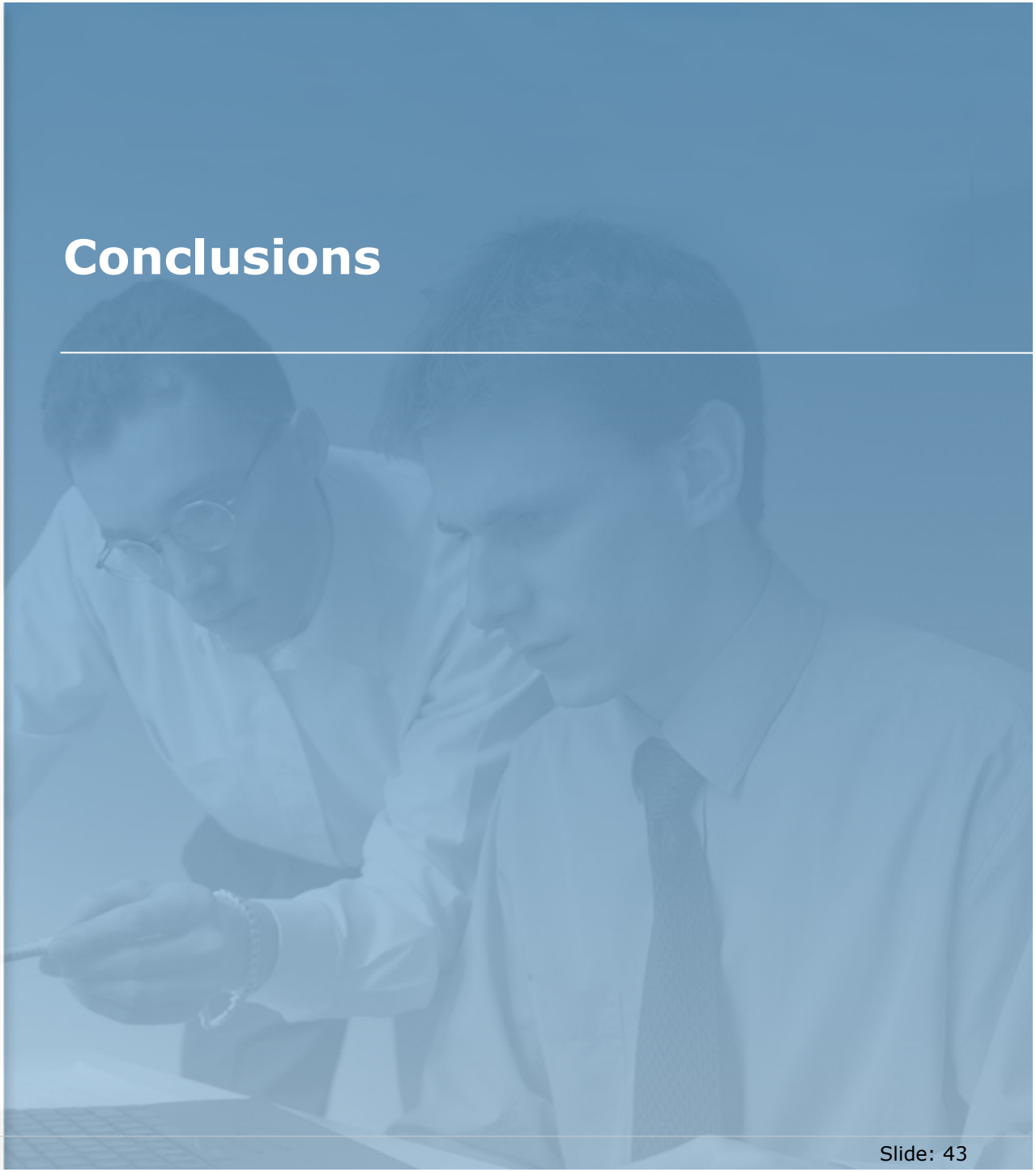
Scenario: Add two numbers
Given the calculator has just been powered on
When I press the button "1"
And I press the button "+"
And I press the button "2"
And I press the button "="
Then I should see "3" on the screen

Scenario: Subtract two numbers
...

http://www.open-do.org/projects/xreq

AdaCore
The GNAT Pro Company

Conclusions



Conclusions

- **AdaCore committed to**
  - Safety high-integrity development
  - Open development
  - Looking for new ways of helping Ada and safety-critical developers

- **Open-DO**
  - Lean and Agile methodologies
  - For safety-critical development
  - Open source tools and material
    - Including certification/qualification material

www.open-do.org