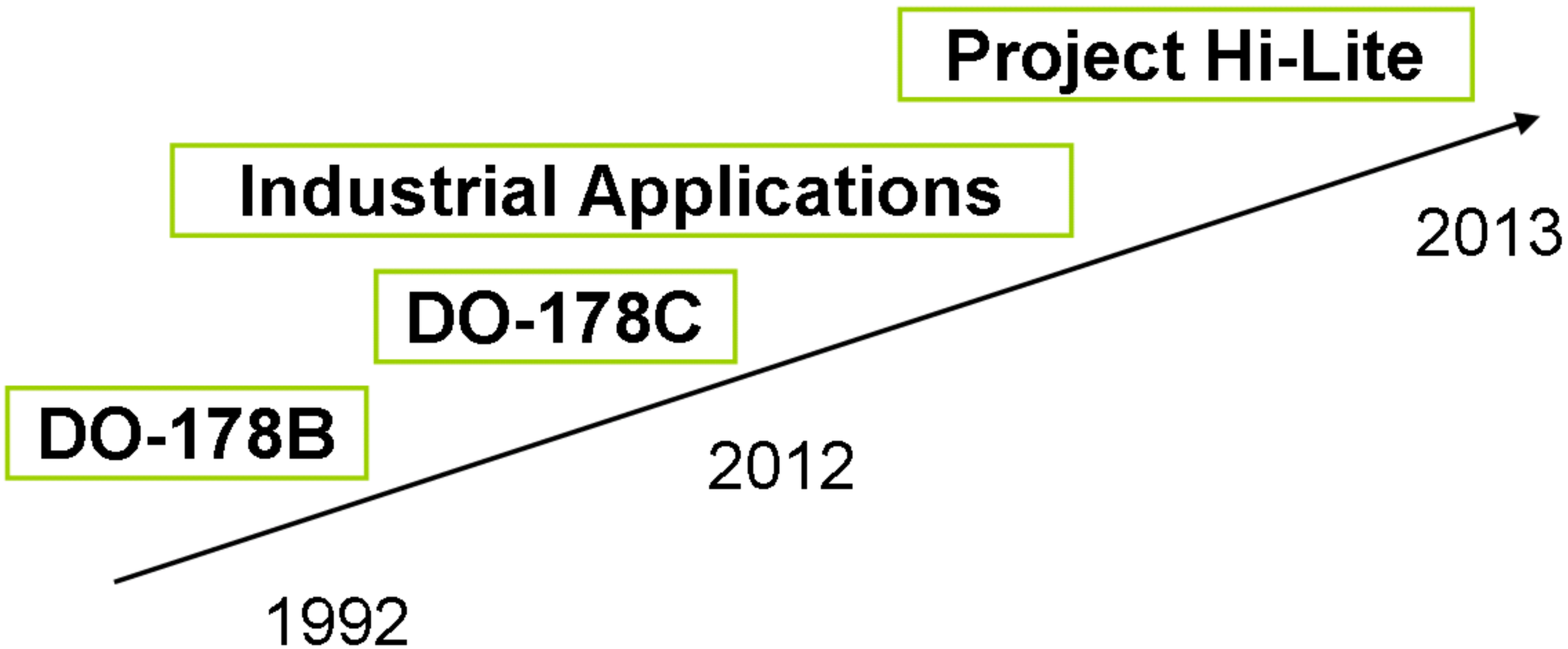
Formal Verification in Aeronautics: Current Practice and Upcoming Standard

Yannick Moy, AdaCore
ACSL Workshop, Fraunhofer FIRST

AdaCore
The GNAT Pro Company

HI-LITE

Outline


Project Hi-Lite
Industrial Applications
DO-178C
DO-178B
2013
2012
1992

DO-178B

What is DO-178B/ED-12B?

RTCA

EUROCAE

Software Considerations in Airborne
Systems and Equipment Certification

What is DO-178B/ED-12B?

RTCA

EUROCAE

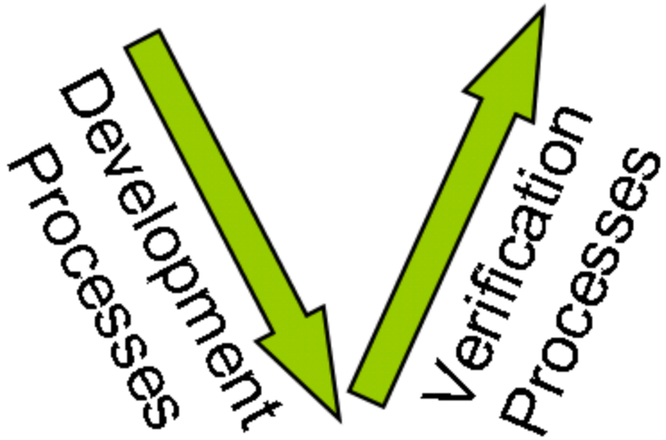
Software Considerations in Airborne
Systems and Equipment Certification

System Aspects

What is DO-178B/ED-12B?

RTCA  EUROCAE

Software Considerations in Airborne
Systems and Equipment Certification

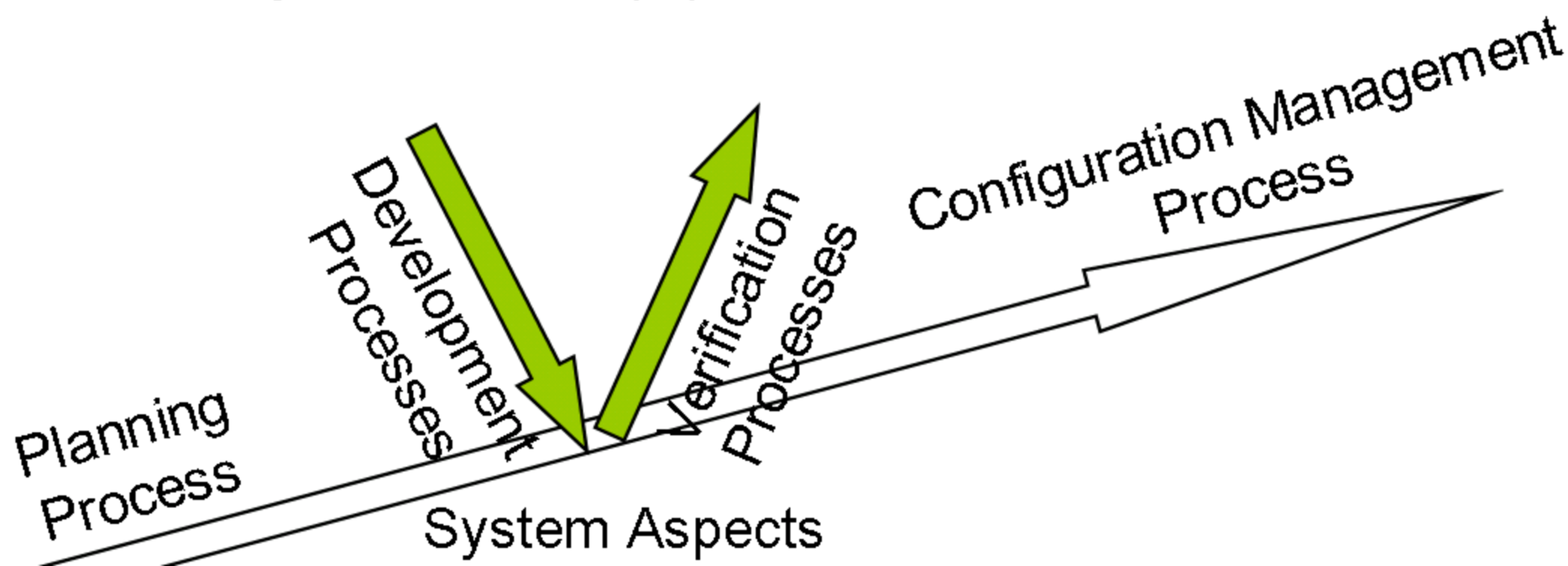


System Aspects

What is DO-178B/ED-12B?

RTCA  EUROCAE

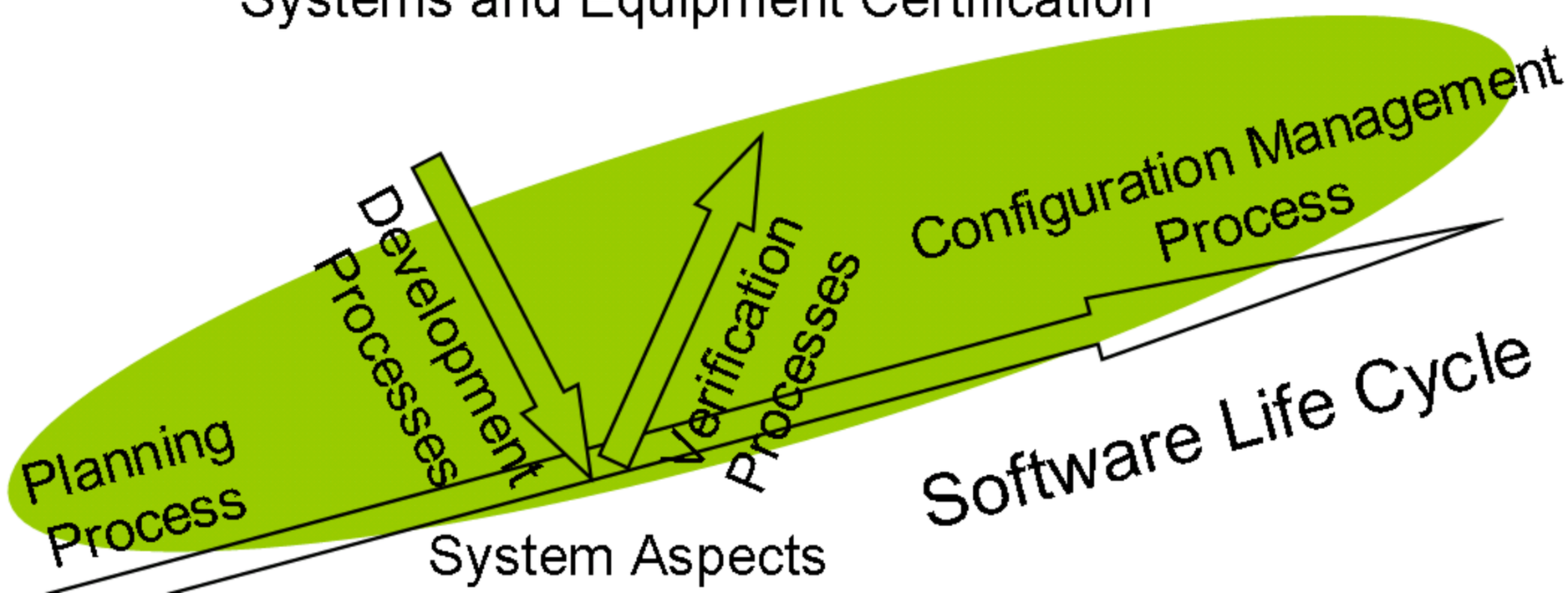
Software Considerations in Airborne
Systems and Equipment Certification



What is DO-178B/ED-12B?

RTCA  EUROCAE

Software Considerations in Airborne
Systems and Equipment Certification



Automatic Verification by Testing

Reviews and Analyses

Testing Process

Verification is not simply testing.Testing, in general, cannot show the absence of errors.

Automatic Verification by Testing

Reviews and Analyses

Testing Process

EVERYTHING from:

- Development process
- Verification process

Mostly manual

Review = inspection
Analysis = evidence

Automatic Verification by Testing

Reviews and Analyses

EVERYTHING from:

- Development process
- Verification process

Mostly manual

Review = inspection
Analysis = evidence

Testing Process

Requirements-based

- Normal range
- Robustness

Test coverage

- Statement
- Branch
- MC/DC

Automatic Verification by Testing

Reviews and Analyses

EVERYTHING from:

- Development process
- Verification process

Mostly manual

Review = inspection
Analysis = evidence

Testing Process

Requirements-based

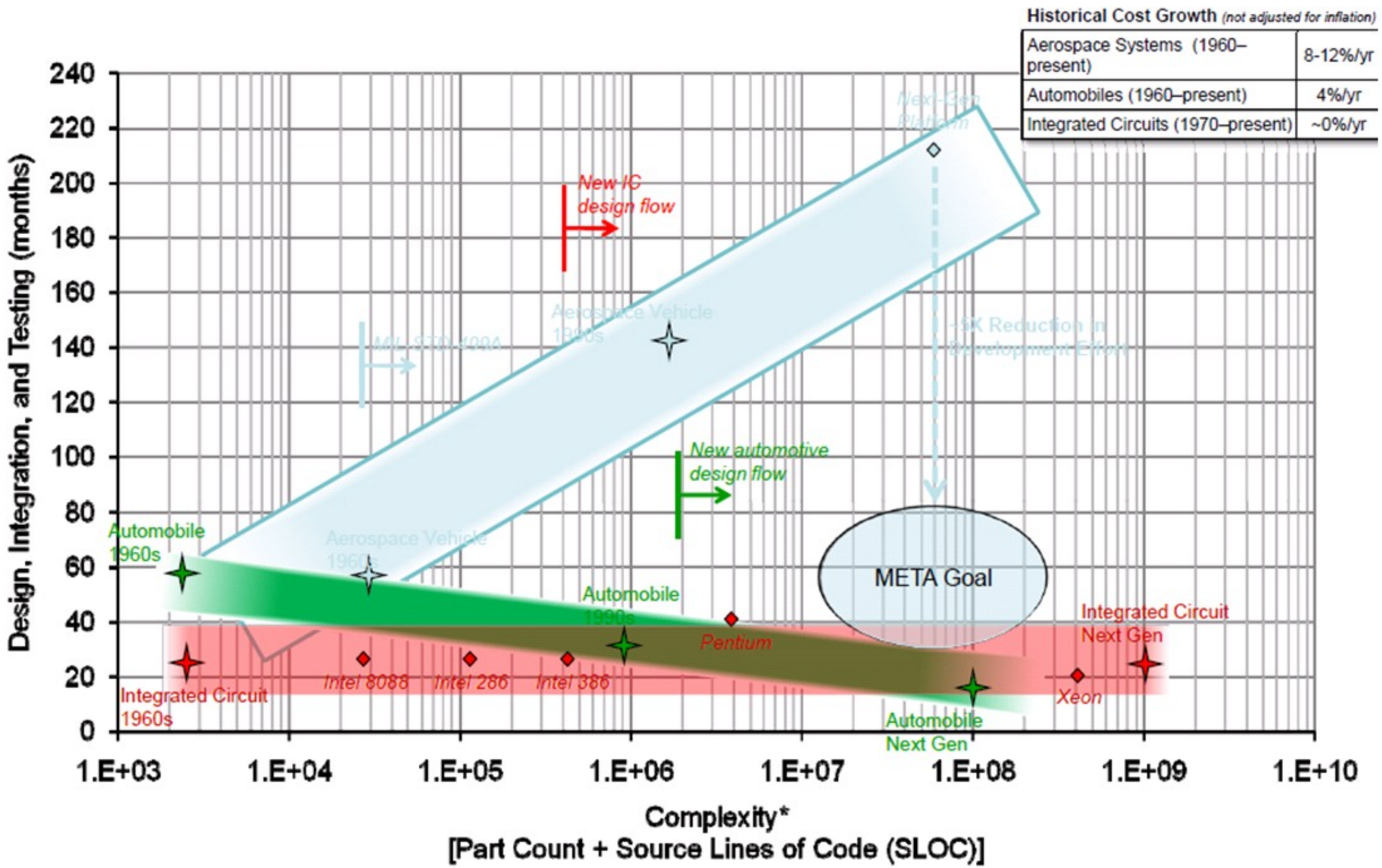
- Normal range
- Robustness

Test coverage

- Statement
- Branch
- MC/DC

Cost is Driving Interest for *Correct-by-Construction*

Kirstie Bellman, META project, Boeing



Historical Cost Growth *(not adjusted for inflation)*

Aerospace Systems (1960–present)	8-12%/yr
Automobiles (1960–present)	4%/yr
Integrated Circuits (1970–present)	~0%/yr

Formal Methods

One page in DO-178B

Formal methods are complementary to testing.

Reserved to problems which cannot be tested: *concurrency, distributed processing, redundancy management and synchronization*

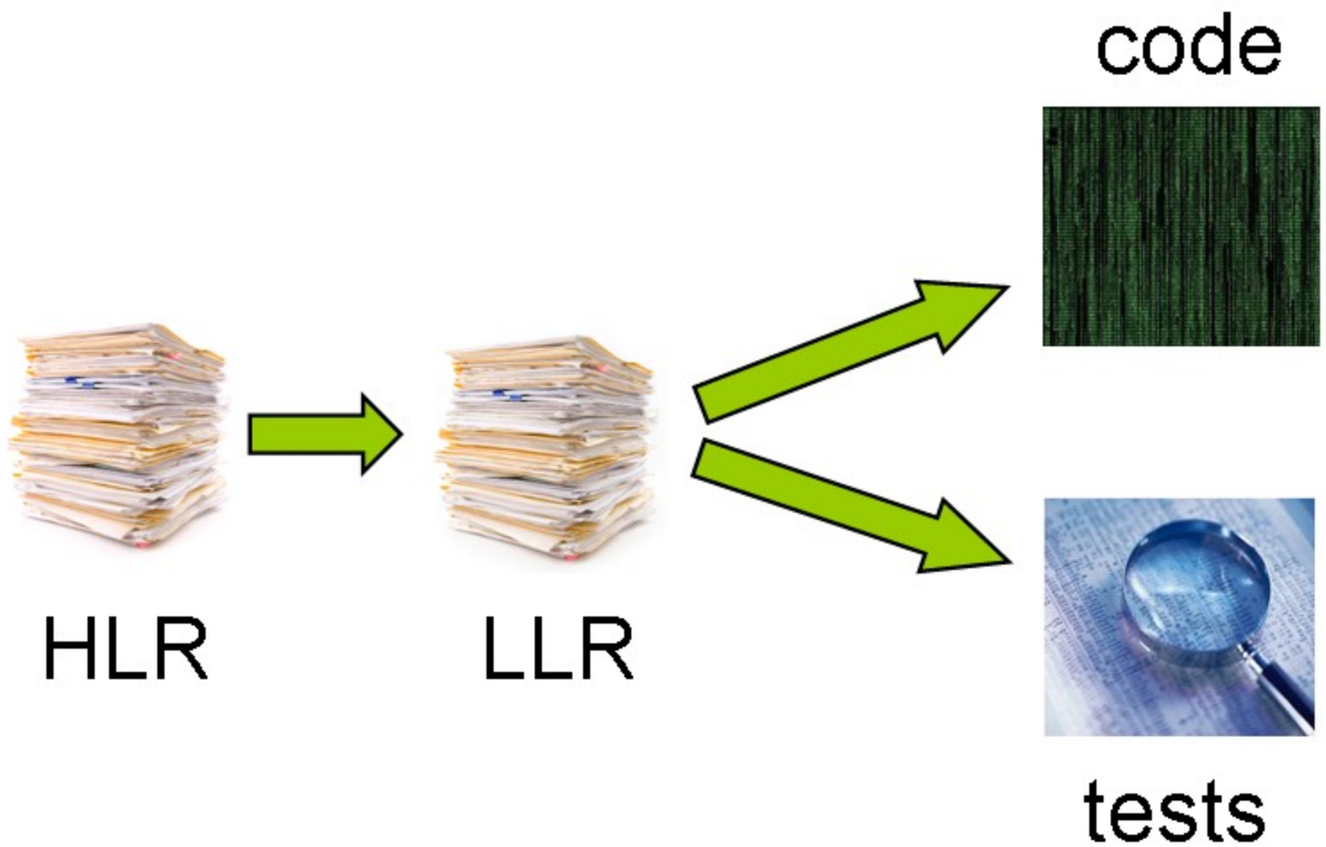
The use of formal specifications alone forces requirements to be unambiguous.

Most Important Lesson

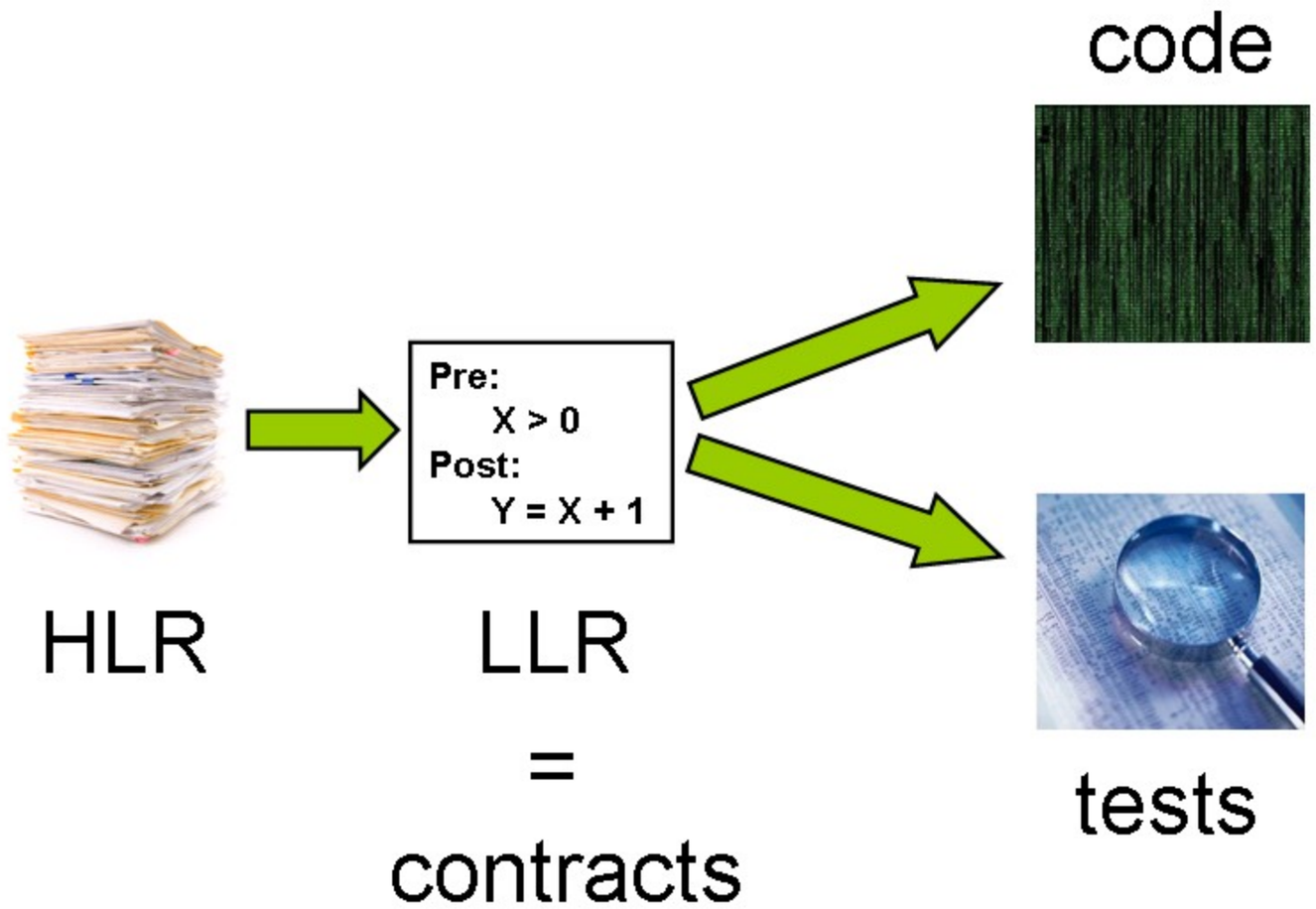
Most of the errors in software development are now generally accepted as being attributable to errors in requirements (LLR or HLR).

FM Discussion Paper

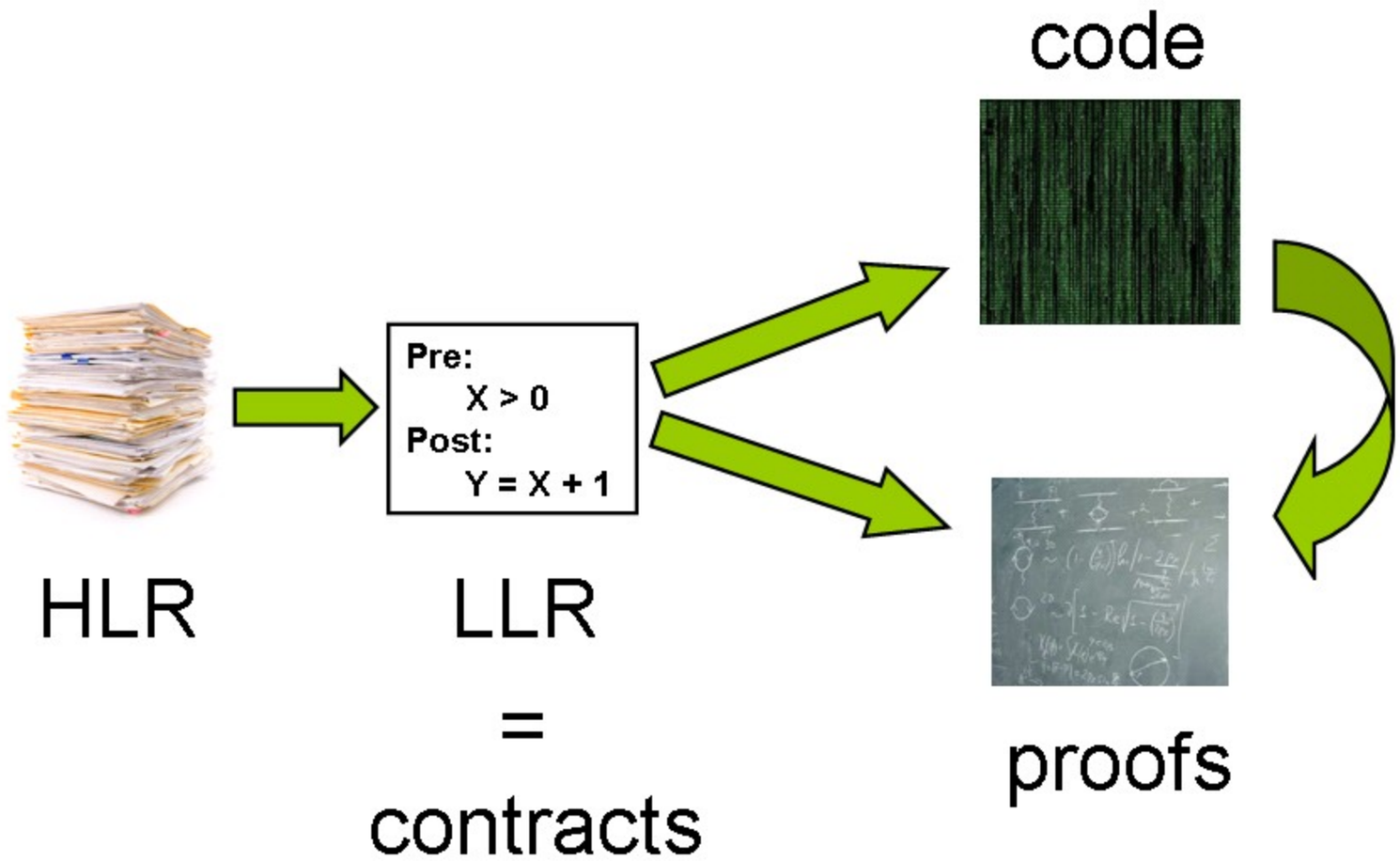
Airbus Pioneering Work



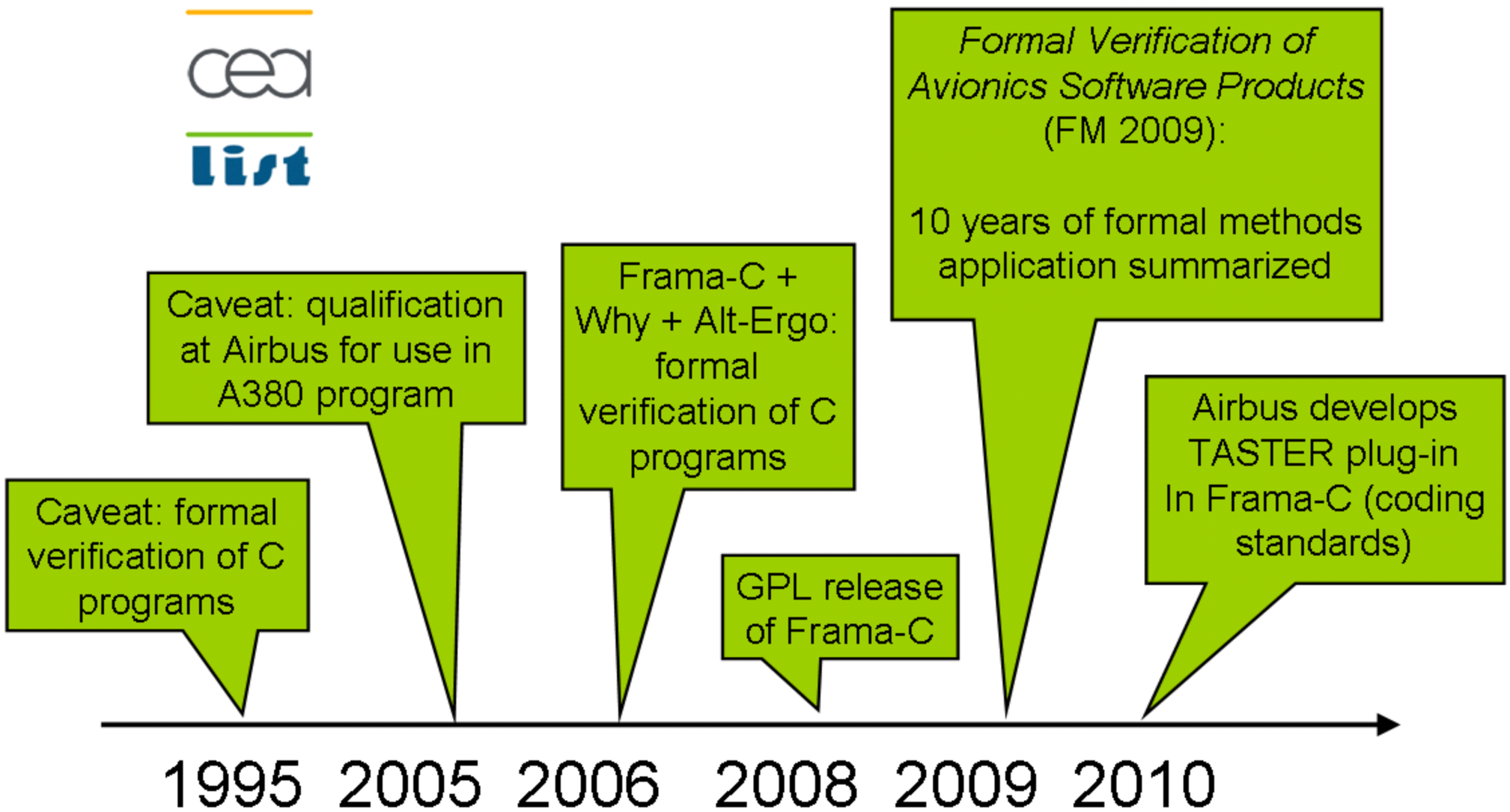
Airbus Pioneering Work



Airbus Pioneering Work



Airbus / CEA LIST Joint Work



DO-178C

Acknowledging Modern Development Techniques

DO-178C

Acknowledging Modern Development Techniques

DO-178C

+ 

Tools qualification document

+ 

Model based development supplement

+ 

Object-oriented & related tech. supplement

+ 

Formal methods supplement

Acknowledging Modern Development Techniques

DO-178C

+ 

Tools qualification document

+ 

Model based development supplement

+ 

Object-oriented & related tech. supplement

+ 

Formal methods supplement

Formal Methods Supplement

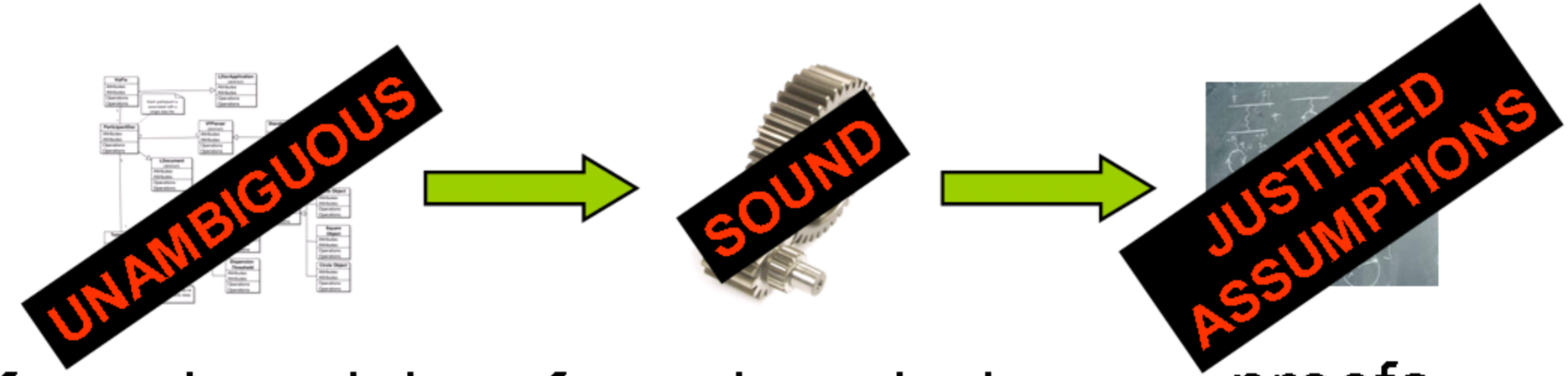
Formal methods […] might be the primary source of evidence for the satisfaction of many of the objectives concerned with development and verification.

formal model → formal analysis → proofs

deductive methods
model checking
abstract interpretation

Formal Methods Supplement

Formal methods […] might be the primary source of evidence for the satisfaction of many of the objectives concerned with development and verification.



formal model

formal analysis

deductive methods
model checking
abstract interpretation

proofs

Benefits of Formal Modeling

FM Supplement & FM Discussion Paper

Improve requirements

Unambiguous description of requirements

Precise communication between engineers

Reduce error introduction

Objective verification evidence:

- one formal model (consistency and accuracy)
- between formal models (compliance)

Benefits of Formal Analysis

FM Supplement & FM Discussion Paper

Improve error detection

Detect exceptions and deadlocks

Detect unintended functions (wrongdoing)

Reduce effort

Non-interference (MILS)

WCET / bounded stack size

Correct (a)synchronous behavior

Relation to Coverage

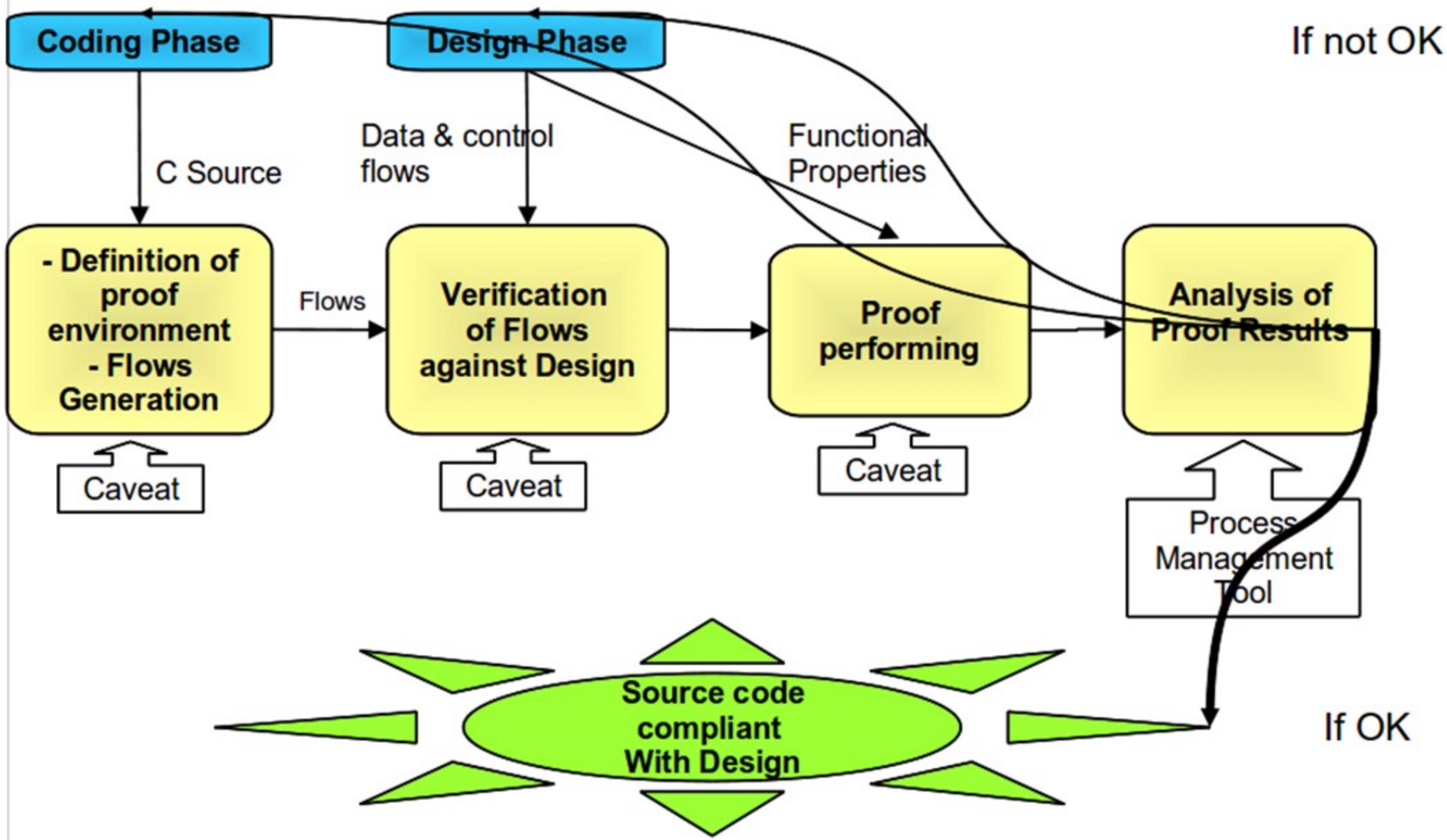
Structural coverage analysis, however, is driven out of the impracticality of achieving exhaustive testing.

When only formal methods are used […] alternative activities are required for coverage analysis.

FM Supplement

Industrial Examples

Airbus – Unit Proof



Airbus – Example of Property

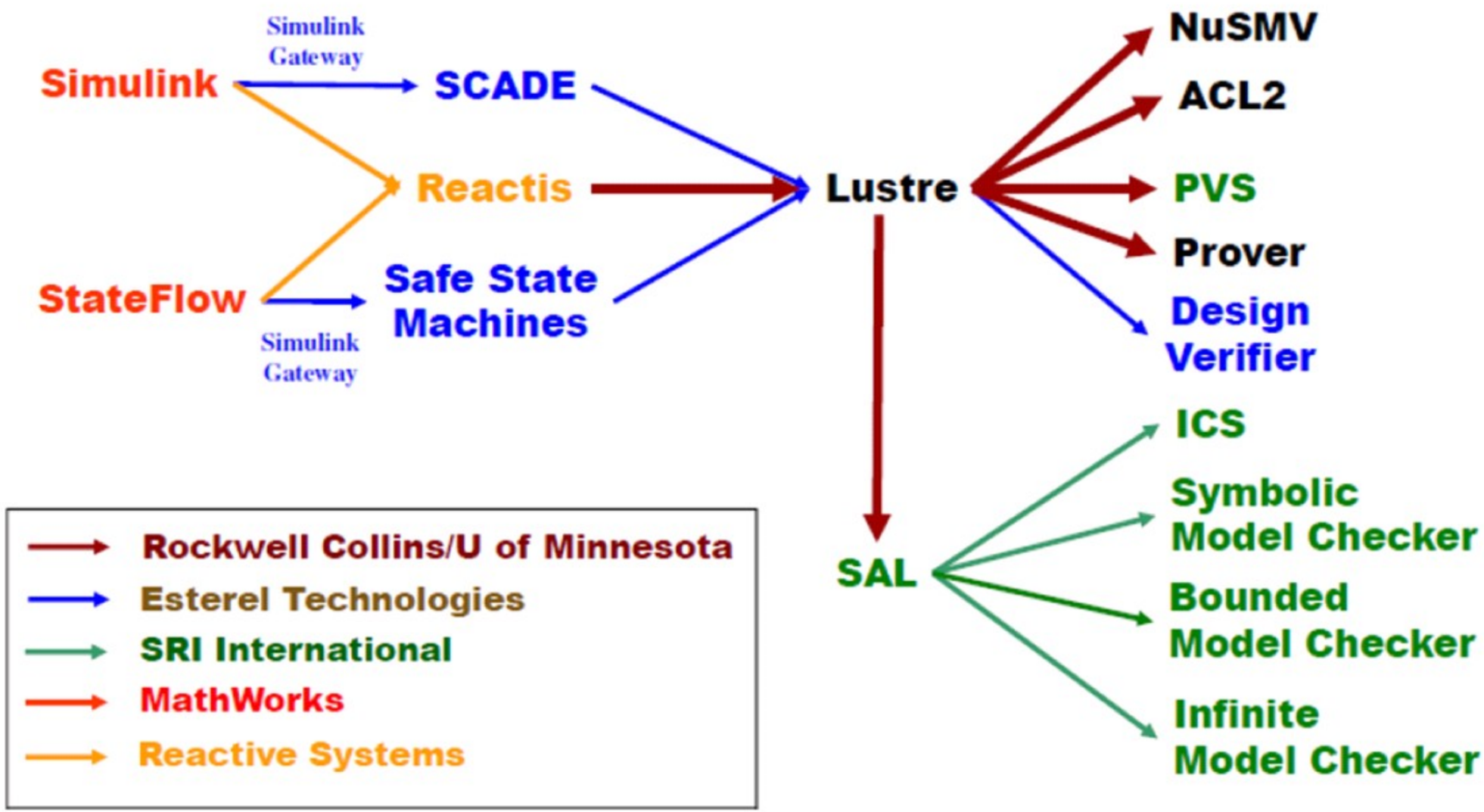
```
LET COND_FCT = (
    ∀ k ∈ int. k>0 and k<=A1F2_ZONE_SIZE ⇒
                        (A1F2_Memory_Zone[k] = 0xFF) );
```

The initial value is correct for all the indexes

```
LET COND_ERR = (
    ∃ k ∈ int. k>0 and k<=A1F2_ZONE_SIZE and
                        (A1F2_Memory_Zone[k] ≠ 0xFF) );
```

There exists an index for which the initial value is wrong

Rockwell-Collins – Model-Based Software Design



Rockwell-Collins – Example of Property

AG(LEFT_DU_AVAILABLE -> LEFT_DU_APPLICATION != BLANK)
AG(RIGHT_DU_AVAILABLE ->RIGHT_DU_APPLICATION != BLANK)

In all reachable states, if the left DU is available, then its application shall not be blank.

AG(LEFT_DU_APPLICATION != MAP ->
AX(LEFT_DU_APPLICATION = MAP ->
CURSOR_LOCATION = LEFT_DU))

In any state which sets LEFT_DU_APPLICATION to MAP, the CURSOR_LOCATION must be LEFT_DU.

SHOLIS – Separation, Contracts and Run-time Errors

Z specification

SPARK code + specification

Rigorous argument

150 proofs
500 pages

SPARK Examiner

SPARK Simplifier

9000 VCs

SPARK Checker

SHOLIS (well, Tokeneer) – Example of Property

```
procedure AddElementToLogFile
   (ElementID : in ElementT; Description : in DescriptionT);

      --# global in       Clock.Now;
      --#        in out NumberLogEntries;
      --# derives AuditSystemFault,
      --#          LogFiles          from *,
      --#                                Description &
      --#          NumberLogEntries from *;

      --# pre  NumberLogEntries < MaxLogEntries;
      --# post NumberLogEntries = NumberLogEntries~ + 1 and
      --#       (LogFileEntries~(CurrentLogFile~) = Max ->
      --#          LogFileEntries(CurrentLogFile) = 1;
```

Project Hi-Lite

Perceived Limitations of Formal Methods

today for…

Software

state-space explosion
alien to engineers
theory versus reality

FM Discussion Paper

Perceived Limitations of Formal Methods

today for…

Software

state-space explosion
alien to engineers
theory versus reality

FM Discussion Paper

not long ago for…

Hardware

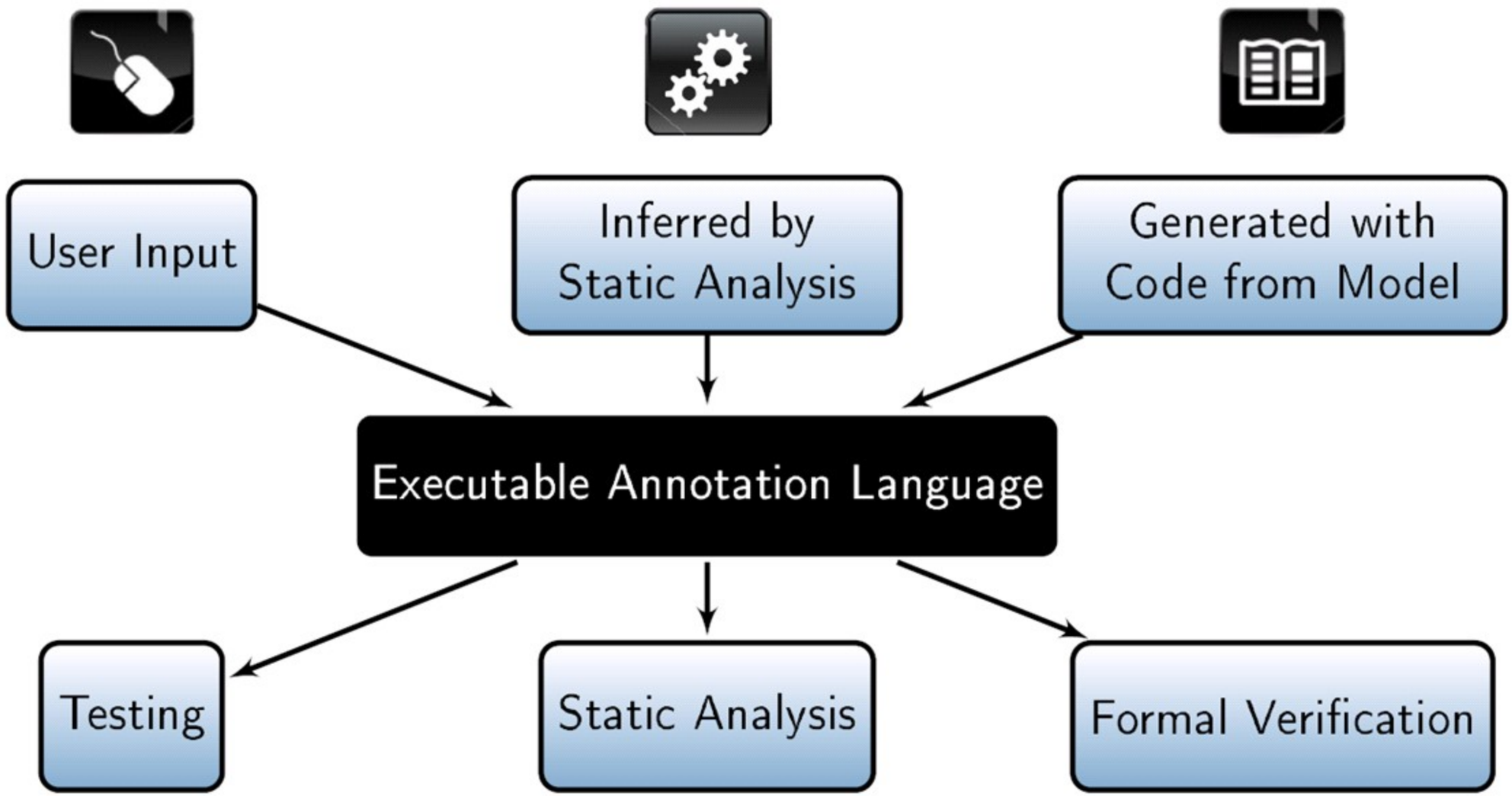
capacity limited
difficult to use
lacking methodologies

SCDsource
Special Technology Report

Limitations of Unit Proof vs Unit Testing

- **Expertise**: required for writing contracts and carrying proof
- **Duplication**: “contract” not shared between testing and proof
- **Isolation**: unit test and unit proof cannot be combined
- **Confusion**: not the same semantics for testing and proof
- **Debugging**: contracts and proof cannot be executed

Solution: Executable Contracts



Executable Contracts in Ada 2012

```
function Sqrt (X : Integer) return Integer with
  Pre  => X >= 0,
  Post => Sqrt'Result >= 0 and then
          Sqrt'Result ** 2 <= X and then
          (Sqrt'Result + 1) ** 2 > X,
  Test_Case => (Name     => "test case 1",
                Requires => X = 100,
                Ensures  => Sqrt'Result = 10),
  Test_Case => (Name     => "robustness test case",
                Requires => X = -1,
                Ensures  => Sqrt'Result = 0);
```

Executable Contracts in Ada 2012

```
function Sqrt (X : Integer) return Integer with
  Pre  => X >= 0,
  Post => Sqrt'Result >= 0 and then
          Sqrt'Result ** 2 <= X and then
          (Sqrt'Result + 1) ** 2 > X,
  Test_Case => (Name     => "test case 1",
                Requires => X = 100,
                Ensures  => Sqrt'Result = 10),
  Test_Case => (Name     => "robustness test case",
                Requires => X = -1,
                Ensures  => Sqrt'Result = 0);
```

Executable Contracts in Ada 2012

```
function Sqrt (X : Integer) return Integer with
  Pre  => X >= 0,
  Post => Sqrt'Result >= 0 and then
          Sqrt'Result ** 2 <= X and then
          (Sqrt'Result + 1) ** 2 > X,
Test_Case => (Name     => "test case 1",
              Requires => X = 100,
              Ensures  => Sqrt'Result = 10),
Test_Case => (Name     => "robustness test case",
              Requires => X = -1,
              Ensures  => Sqrt'Result = 0);
```

Executable Contracts in Ada 2012

```
function Sqrt (X : Integer) return Integer with
  Pre  => X >= 0,
  Post => Sqrt'Result >= 0 and then
          Sqrt'Result ** 2 <= X and then
          (Sqrt'Result + 1) ** 2 > X,
Test_Case => (Name     => "test case 1",
              Requires => X = 100,
              Ensures  => Sqrt'Result = 10),
Test_Case => (Name     => "robustness test case",
              Requires => X = -1,
              Ensures  => Sqrt'Result = 0);
```

Executable Contracts in Ada 2012

```
function Sqrt (X : Integer) return Integer with
  Pre  => X >= 0,
  Post => Sqrt'Result >= 0 and then
          Sqrt'Result ** 2 <= X and then
          (Sqrt'Result + 1) ** 2 > X,
Test_Case => (Name     => "test case 1",
              Requires => X = 100,
              Ensures  => Sqrt'Result = 10),
Test_Case => (Name     => "robustness test case",
              Requires => X = -1,
              Ensures  => Sqrt'Result = 0);
```

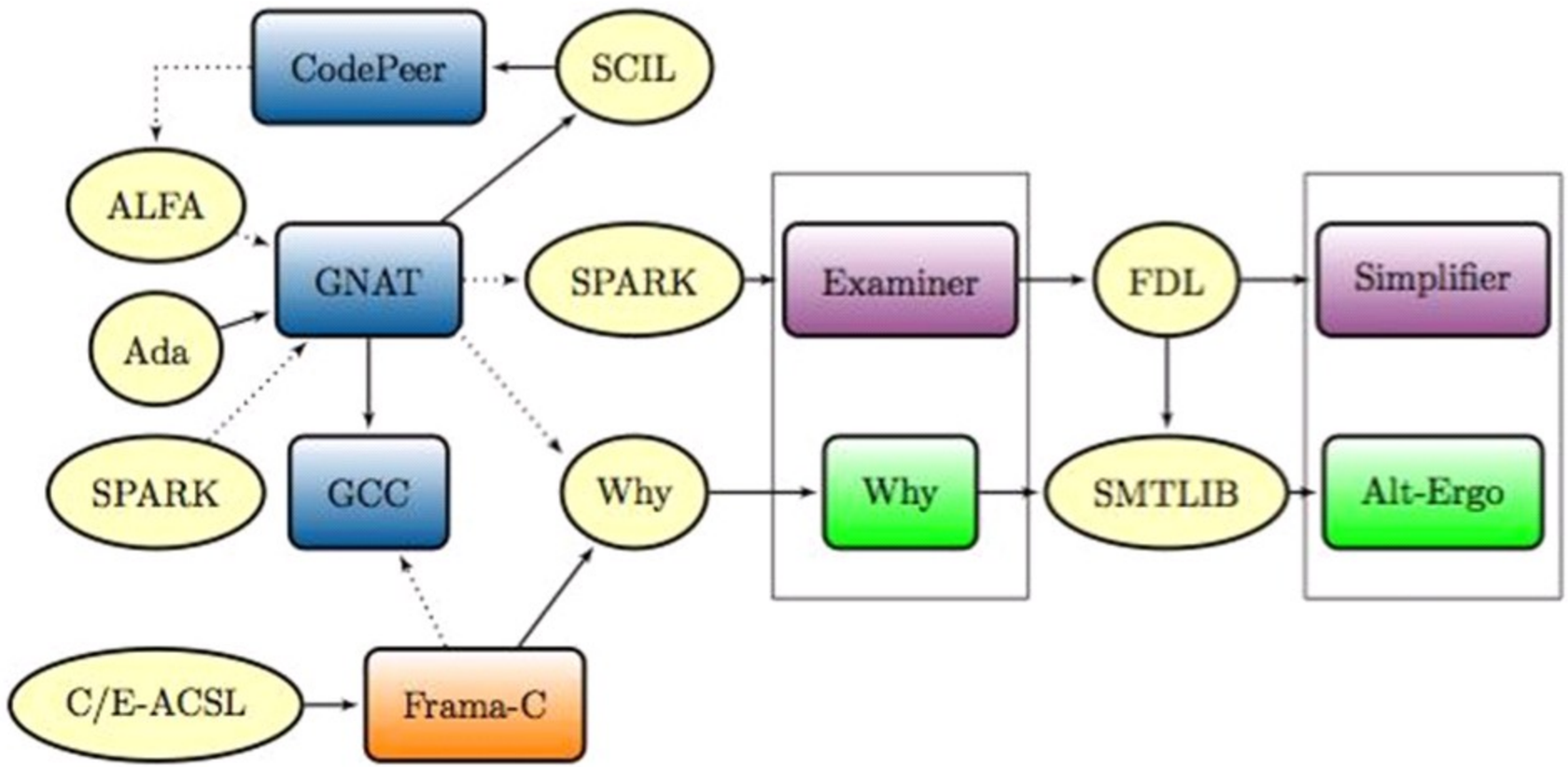
Objectives of Project Hi-Lite

- Facilitate proofs of safety / security / properties
- One language of assertions
- Testing + proof + static analysis
- Subset of C or Ada code
- Mixed Ada-C code

Project Partners

Project Partners

Sketch of Communications Between Tools



Open Project

	FLOSS	Latest Open Release	Open BTS	Open mailing-list	Open VCS	Open dev.
SPARK	GPL	2010	GAP	one-way		2 teams
GCC (GNAT)	GPL	2010				
GPS	GPL	2010	GAP			
CodePeer	GPL					
Why / Alt-Ergo	GPL / CeCILL-C	2010				
Frama-C	LGPLv2	2010				plug-ins
Hi-Lite	GPL	2011?				

Conclusions

While DO-178B was centered on testing, DO-178C allows formal verification instead of testing.

Past industrial applications have shown formal verification can be cost-effective.

To bridge the gap between unit proof (FM) and unit testing (engineers), project Hi-Lite defines executable contracts for C and Ada.