Aero Engine Controls

A Rolls-Royce plc and Goodrich Corporation joint venture

ED-218 – The Formal Methods Supplement for ED-12C

Duncan Brown
Chief of Systems Capability
Aero Engine Controls

People you can count on, products you can trust.

Overview

- **Readiness for formal methods**
- **Intro to the Formal Methods Supplement**
  - **formal methods from a DO-178/ED-12 perspective**
  - **verifying to DO-178/ED-12 with formal analysis**
  - **additional objectives when using formal methods**
  - **Some examples of current industry practice**
  - **Benefits**
- **Summary**

Aero Engine Controls

1968 NATO Conference

"There are, fundamentally, two different methods of determining whether a product meets its specification.

- One can analyse the product in great detail and from this determine if it is in accordance with its specification, or
- one can measure its performance experimentally and see if the results are in accord with the specification; the number and sophistication of the experiments can be varied to provide the degree of confidence required of the results.

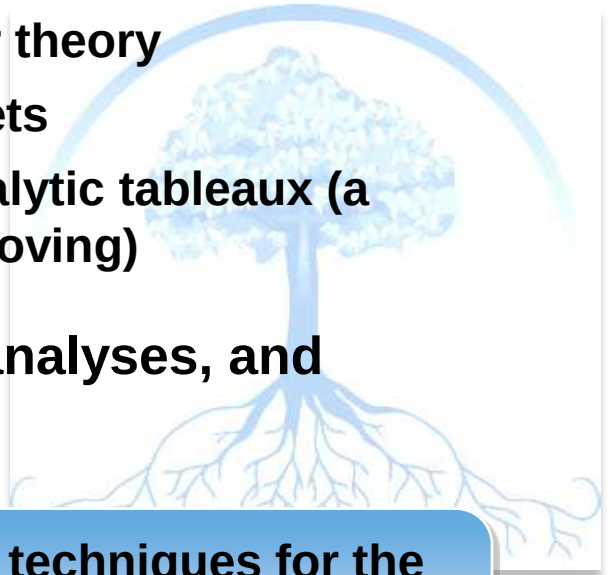
Our current software appraisals are biased towards the analytic approach but the limited time and effort available and the often limited access to detailed information prevent the appraisal doing away with the need for an experimental approach to software testing."

From the 1968 NATO Software Engineering Conference that introduced the phrase "software engineering". Reformatted from a working paper "The Testing of Computer Software" by A.I. Llewelyn and R.F. Wickens.

Aero Engine Controls

Analytic Approaches

- **Analytic: Having the ability to analyze; or the process of breaking a concept down into more simple parts, so that its logical structure is displayed**
- **In mathematics, analytic approaches include:**
  - **analytic combinatorics, geometry, and number theory**
  - **in set theory: analytical hierarchy & analytic sets**
  - **in proof theory: analytic proof & method of analytic tableaux (a fundamental concept in automated theorem proving)**
- **Formal methods are rooted in mathematical analyses, and descriptive notations that facilitate analyses**



Formal methods: mathematically based techniques for the specification, development, and verification of software and hardware aspects of digital systems

Analysis in Practice

- Have we come any closer in the 43 years since the NATO conference to realizing a bias ***in practice*** towards analytical – or formal methods in software engineering
  - in place of *or* in cooperation with testing?
- Have we reached a tipping point in the use of formal methods – especially in the aviation industry?

Uphill Battle

"Formal methods are one of the things that the academia keeps pushing even though they don't really work. They're great for writing papers about. They're also completely impractical for real software, but people will keep talking about them for as long as you can make [an] academic career of them ..."

[ref. http://www.reddit.com/comments/k57e/are_formal_methods_worth_the_effort_pdf?sort=new]

Are formal methods really impractical for real software???

Aero Engine Controls

Uphill Battle (2)

"…many research papers are written as if the mathematics were all that matters. They do not show how to relate the formal models and results to the actual code on real machines. Further, they offer no way to deal with the complexity of software systems.

On the other side, most software developers perceive formal methods as useless theory that has no connection with what they do. There is no quicker way to lose the attention of a room full of programmers than to show them a mathematical formula."

[ref. David Parnas. Really Rethinking Formal Methods, January 2010 IEEE Computer]

Is it really useless theory with no connection to real machines?

Aero Engine Controls

Why bother?

"Practitioners have been sceptical about the practicality of formal methods. Increasingly, however, there is evidence that formal methods can yield systems of very high dependability in a cost-effective manner, ….”

[Ref. *Software for Dependable Systems: Sufficient Evidence?* Daniel Jackson, Martin Thomas, and Lynette I. Millett, Editors, Committee on Certifiably Dependable Software Systems, National Research Council, 2007]

Some Success

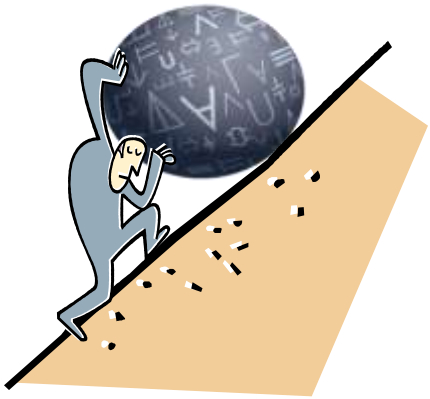
- Some companies are investing in formal methods in their software engineering practice
- Airbus is leading the way for the aviation industry
  - used SCADE for automatic code generation on A340 & A380 to reduce coding errors
  - used formal verification, in place of more conventional methods, on the A380
    - worst case execution time computation of programs (Level A)
    - stack analyzers for stack consumption computation (Levels A, B, & C)
    - proof of absence of run time errors (Level A)
    - unit-level proof of low level requirements in place of unit testing (Level A)
  - continuing use of formal methods in other aircraft programs

More Success

- **Rockwell Collins: developed a suite of translation tools to enable the use of model checking early in design process**
  - **formal analysis of Window Manager logic in their Adaptive Display and Guidance System (ADGS-2100)**
  - **formal analysis of their FCS 5000 flight control mode logic**
    - **translated Simulink models of logic into a formal notation, to allow model checking**
- **National Air Traffic Services (NATS), Interim Future Area Control Tool Support (iFACTS) project to support decision making and facilitate detection of conflicts**
  - **used a formal approach to "reduce risk of integration of new software with existing software"**
    - **used the Z language in the system and software specification**
    - **used SPARK Ada for 87% of the coding**
    - **some run-time exception proofs**

Some Obstacles

- **Readiness concerns: is the aviation industry generally ready for formal methods?**
  - **sufficient expertise**
    - **developers, integrators, certification authorities**
  - **availability of suitable tools**
  - **training**
  - **standardization**

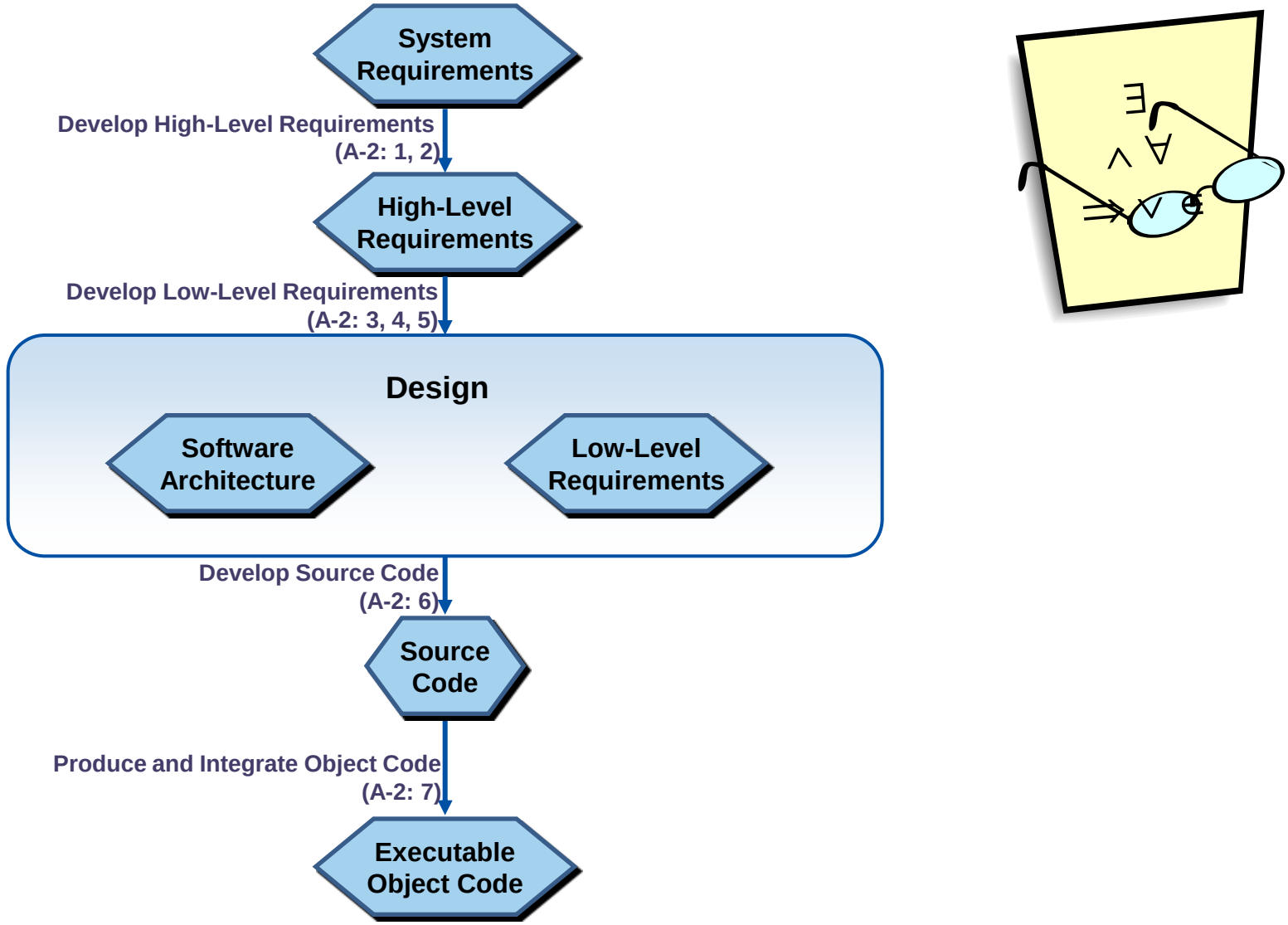


- **Accessibility concerns: are formal methods tools & techniques accessible to the average engineer?**
  - **Do you have to be a specialist/PhD to appropriately use formal methods? Or determine if they have been used appropriately?**
  - **Can an average engineer be given the task of proving correctness of a program/requirement with readily available tools?**

Purpose

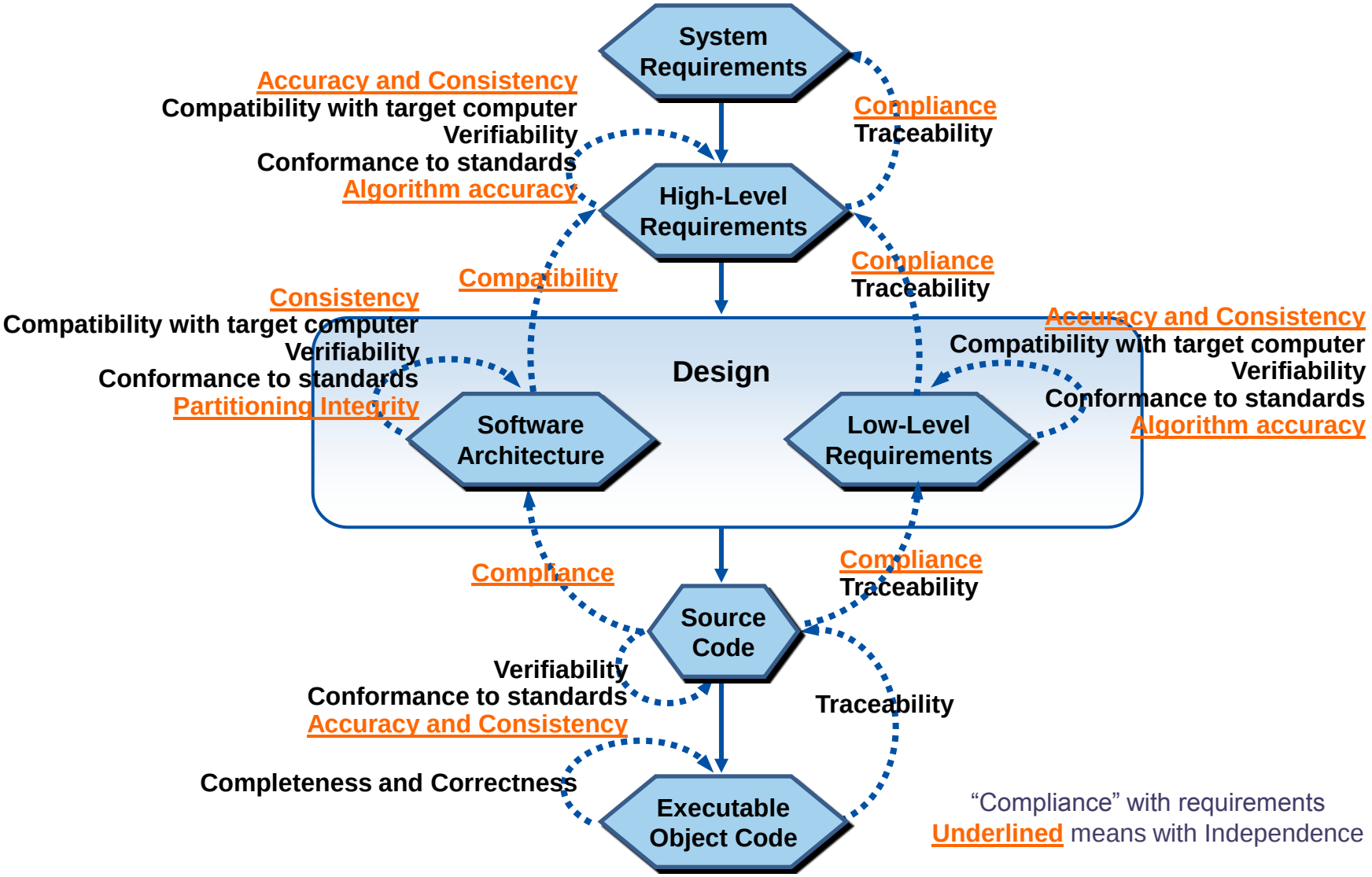
- **The purpose of this talk is not to convince you that formal methods have reached a "tipping point"**
  - **it is to explain the Formal Methods Supplement to DO-178C**
    - **and provide evidence of the usefulness of formal methods in aviation applications today**
- **What the supplement says about using formal methods in a 178 context**
- **What you need to know about formal methods to understand the Formal Methods Supplement**
  - **starting with a view of DO-178 from an analysis perspective**

Development Objectives



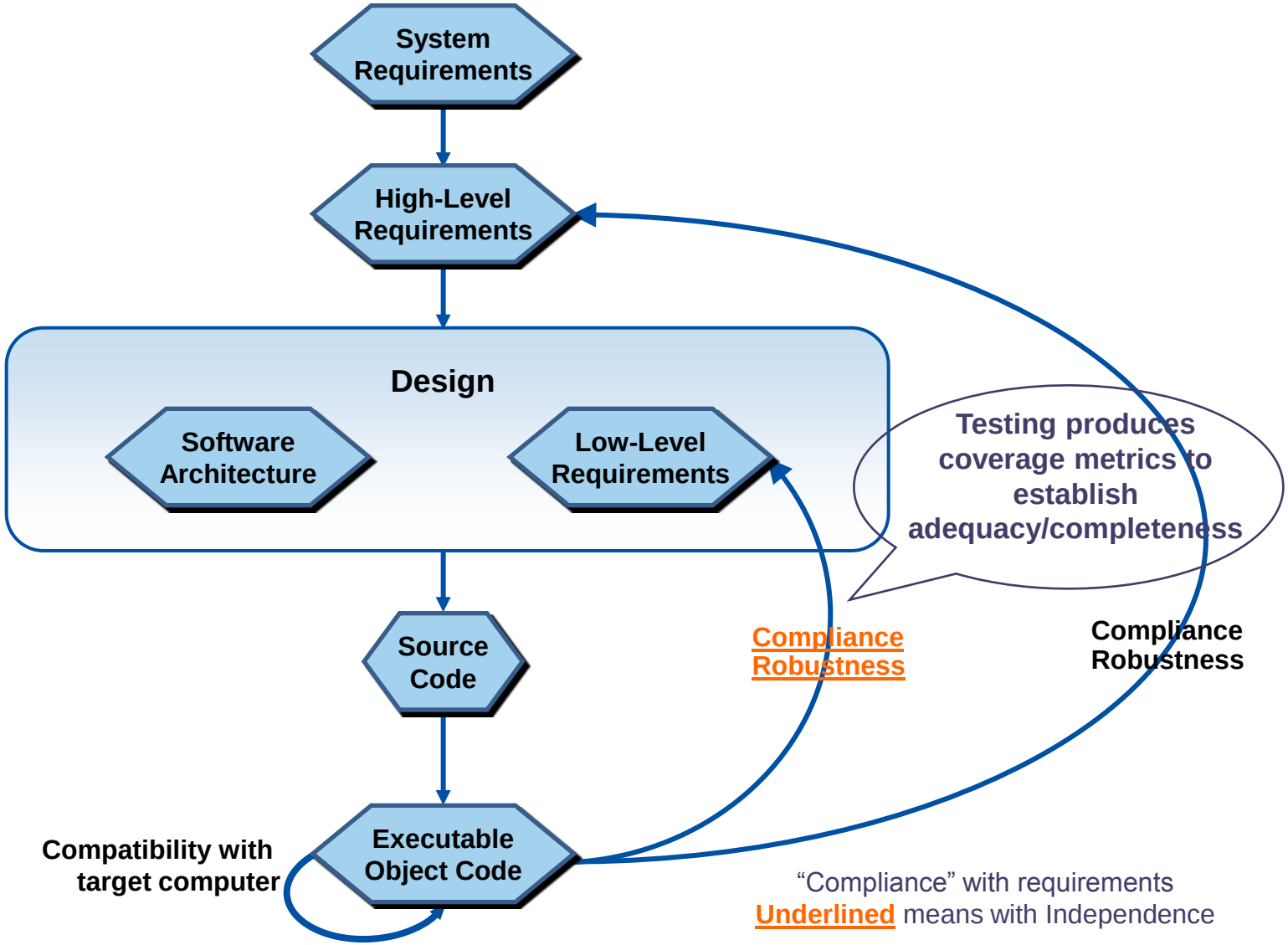
Aero Engine Controls

Verification Objectives: Review & Analysis



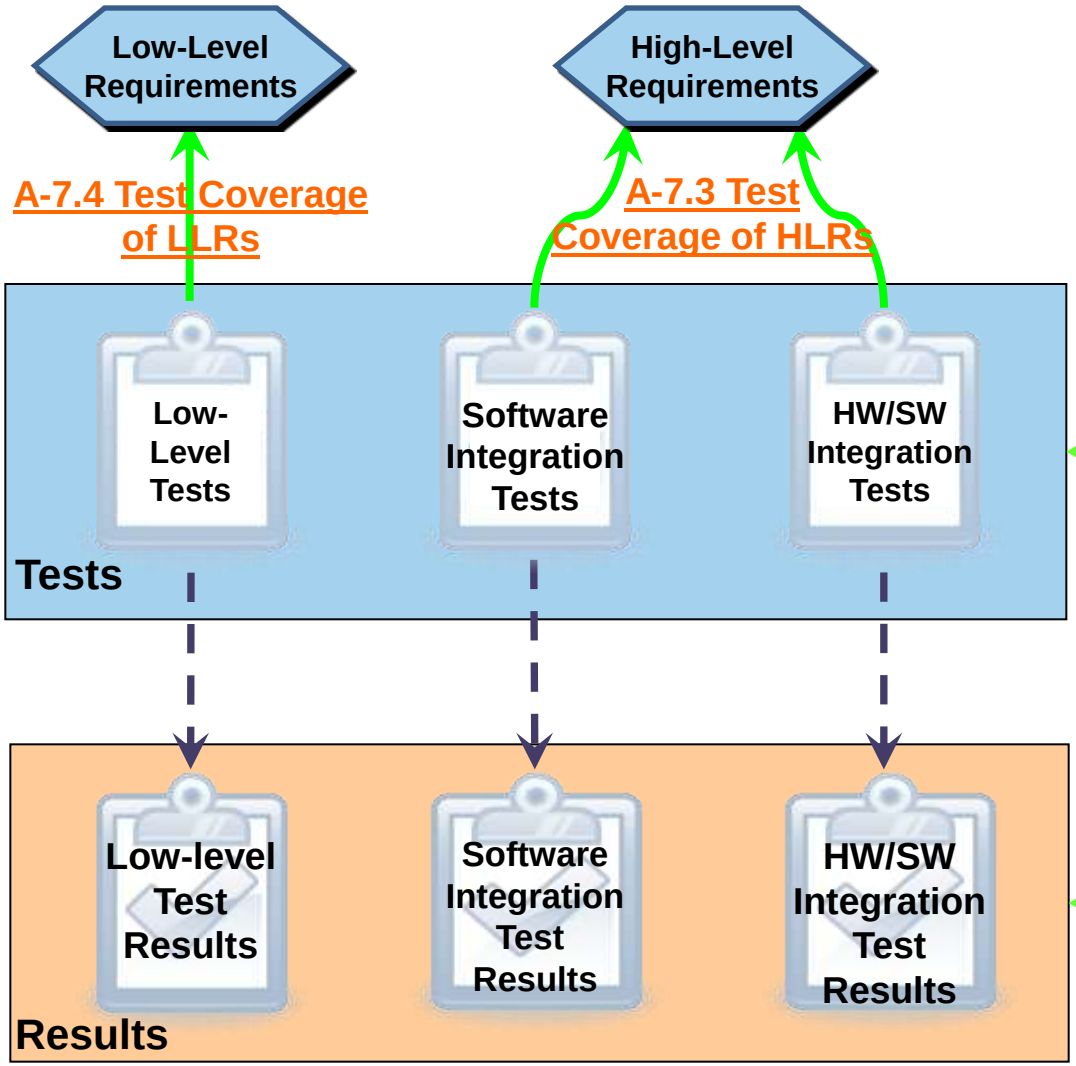
"Compliance" with requirements **Underlined** means with Independence

Verification Objectives: Testing



Aero Engine Controls

Verification Objectives: Verification of Verification Process Results



Formal Methods in 178 Terminology

- **A formal method has 2 components:**
  - **formal analysis that is carried out on a formal model**

Formal Model

Any development artifact can be turned into a Formal Model

- it must be defined using a *formal notation*
  - with precise, unambiguous, mathematically defined syntax and semantics

Formal Analysis

Formal Analysis, applied to a formal model, can be used to meet one or more verification objectives for that artifact

Aero Engine Controls

The Role of the Formal Methods Supplement

- **Formal models can be used without formal analysis to improve and standardize development artifacts**
  - **does not require the use of any supplements**

- **Formal analysis can be used to supplement verification necessary for DO-178C**
  - **does not require the use of any supplements**

- **If formal analysis is used to replace aspects of the DO-178C verification processes, then the Formal Methods Supplement is required**

Aero Engine Controls

Conservative Representation

- **A model is an abstract representation of a given set of aspects of a system**
- **To do formal analysis, the model must be**
  - **written in a formal notation**
  - **a conservative representation of what is being modeled**
    - ***Properties* that hold for the *model* hold for the modeled artifact**
- **This is particularly relevant for a model created as an abstract interpretation and used for timing analysis or Stack usage.**

Note: traditionally this is known as Conservative Approximation but the idea of approximation being acceptable for ED-12C was hard to sell!

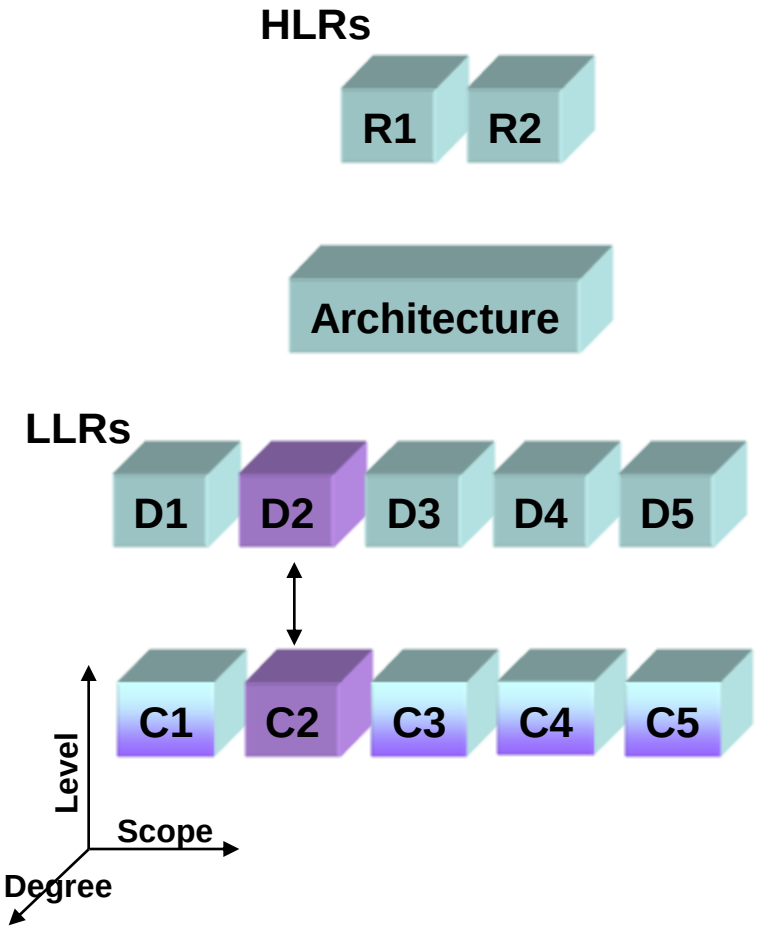
The Importance of being Sound

- **For the Formal Methods Supplement, an analysis method is formal if it has a *sound* mathematical basis, typically given by a formal notation**

> ***A sound method never asserts that a property is true when it may not be true***

Aero Engine Controls

Verification with Formal Analysis



- **Formal Analysis can be applied at various levels**
  - **LLR vs HLR**
  - **Code vs LLR**
  - **Code vs HLR**
  - **Code vs Architecture**
  - **etc.**
- **Their scope can be less than the complete system**
  - **e.g., C2 vs D2 by formal analysis, C1, C3, C4 & C5 by review and test**
- **The "degree" can be less than the full specification**
  - **e.g., all the code can be analysed for worst case time or stack usage; the architecture could be analysed for consistent information flow**

➢ **Formal analysis can be used for most of the verification objectives but can be applied selectively**

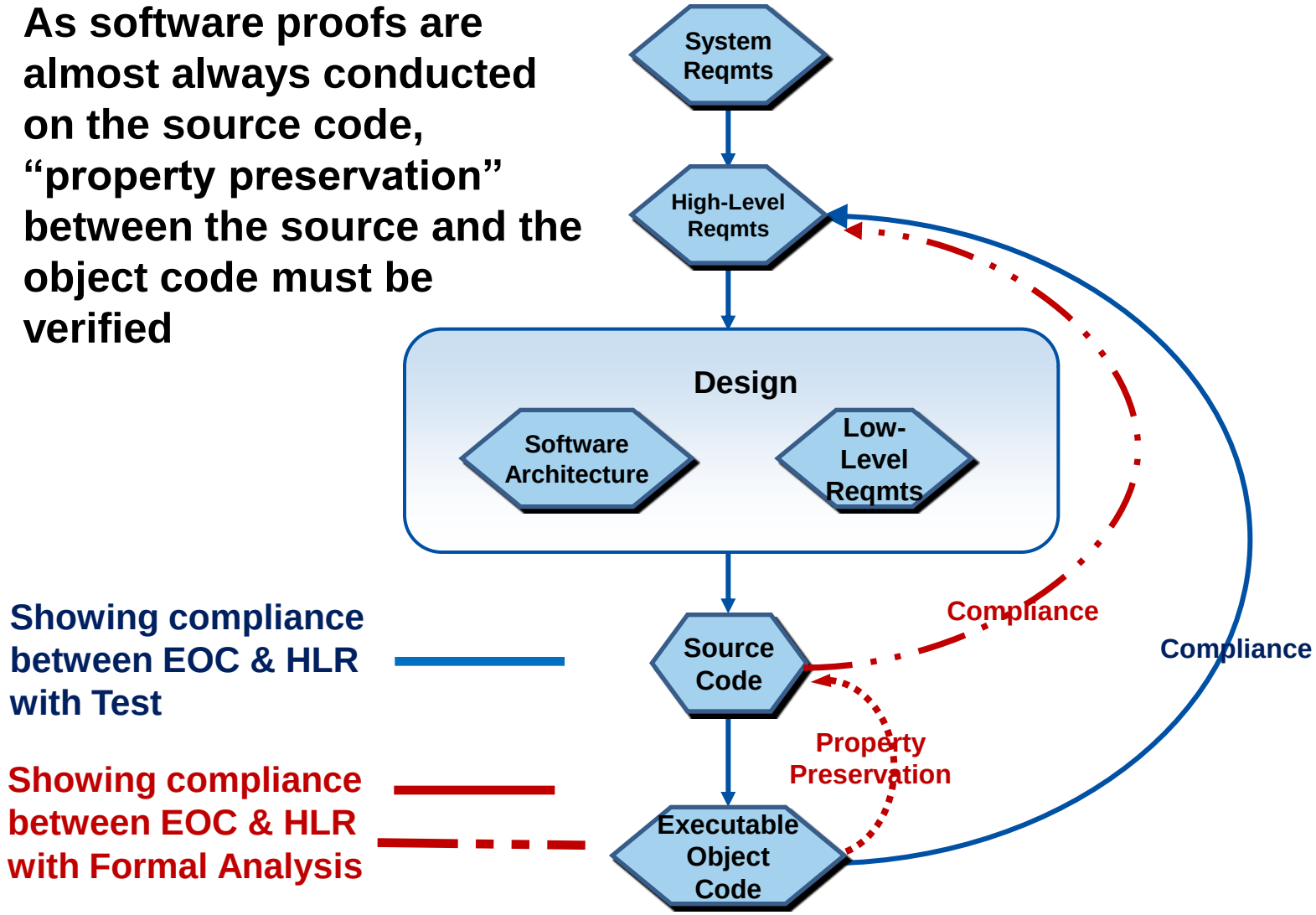
Aero Engine Controls

Substitution for Review and Analysis

- It is important to understand the scope and degree of the formal analysis as any remaining verification must complement the Formal Method and comply with DO-178
- Formal Methods can meet generic DO-178 objectives as follows:
  - Compliance – Prove that an artifact is fully compliant with its requirements
  - Accuracy – Ensure that an artifact is correct and complete with respect to its requirements
  - Consistency – Check that the artifact is self consistent
  - Compatibility – Detect that the artifact conflicts with a target computer specification
  - Verifiability – Once an artifact is formally specified then by definition it is verifiable either by the formal analysis or manually
  - Conformance – Ensures that an artifact conforms to the Formal Notation
  - Traceability – Ensure that all requirements are fully satisfied by the artifact and that everything in the artifact is necessary to satisfy its requirements
  - Algorithm aspects – Check that an artifact meets its specification precisely and under all circumstances
- When a Formal Notation is used, the resultant artifact must be reviewed to ensure that it is a conservative representation of its requirements

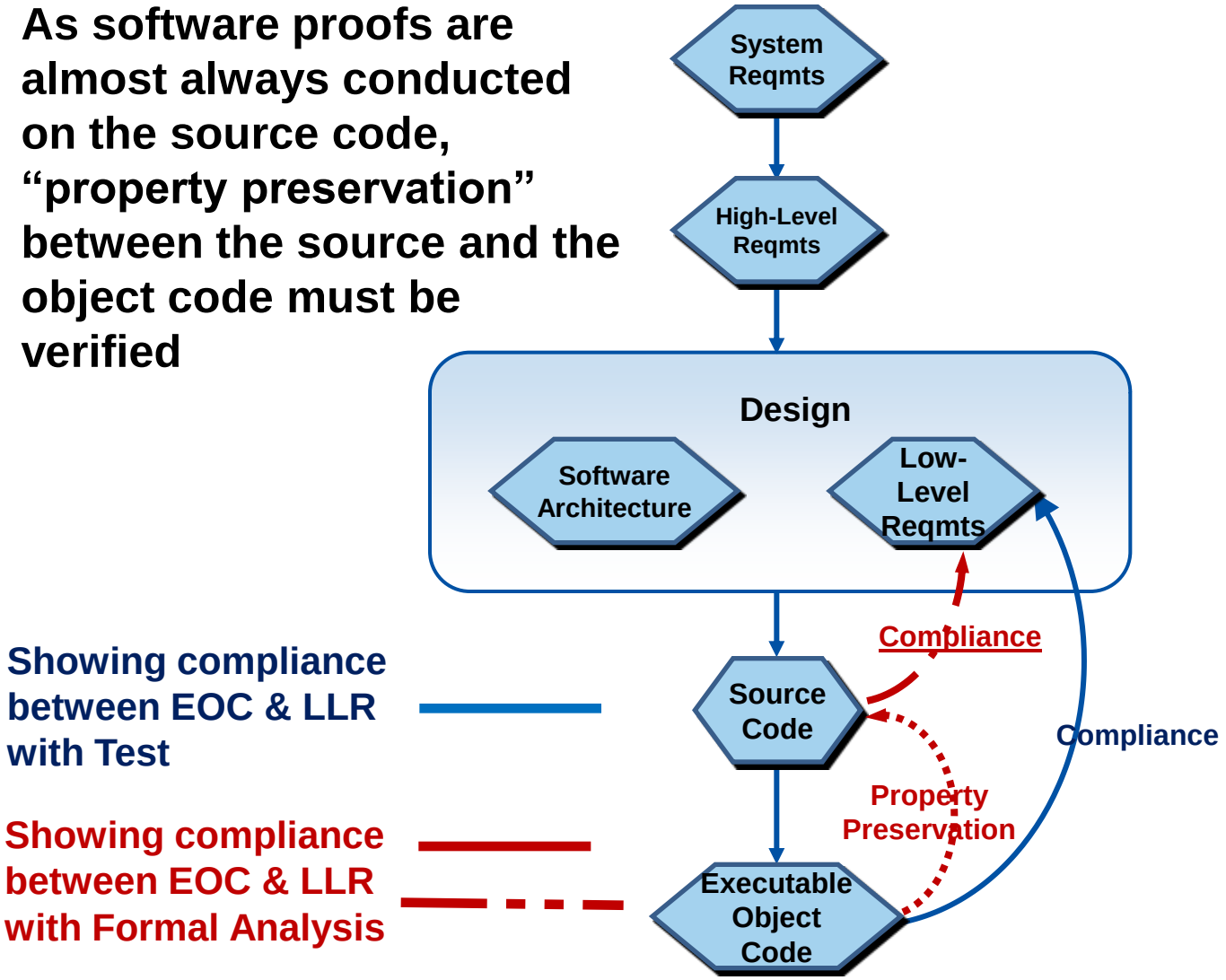
Substitution for Test

- As software proofs are almost always conducted on the source code, “property preservation” between the source and the object code must be verified



Substitution for Test

- As software proofs are almost always conducted on the source code, "property preservation" between the source and the object code must be verified



New Objectives for Formal Analysis

Tables FM.A-3 through FM.A-5 each have 4 new objectives needed when formal analysis is used

Formal analysis cases and procedures are correct.	FM.6.3.6a FM.6.3.6b	●	❍	❍		Software Verification Results	11.14	②	②	②	
Formal analysis results are correct and discrepancies explained.	FM.6.3.6c	●	❍	❍		Software Verification Results	11.14	②	②	②	
Requirements formalization is correct.	FM 6.3i	●	●	❍		Software Verification Results	11.14	②	②	②	
Formal methods is correctly defined, justified, and appropriate.	FM 6.2.1	●	❍	❍	❍	Software Verification Results	11.14	②	②	②	②

Aero Engine Controls

Verification of Verification

DO-178C Annex Table A-7 Objectives	FM Supplement Annex FM.A-7 Objectives
1 - Test procedures correct	FM1 - Formal analysis cases and procedures correct
2 - Test results correct and discrepancies explained	FM2 - Results are correct and discrepancies explained
3 - All high level requirements tested	FM3 - All high level requirements verified
4 - All low level requirements tested	FM4 - All low level requirements verified
5 - 100% MCDC is achieved	FM5-8 - Each requirement is fully verified Set of requirements complete* Unintended data flow detected Dead/deactivated code detected * If this cannot be met then the only alternative is DO-178 structural coverage and therefore, in general, testing.
6 - 100% decision coverage achieved	
7 - 100% statement coverage achieved	
8 - 100% data/control coupling achieved	
	FM9 - Verify property preservation
	FM10 - Formal method is appropriate

Aero Engine Controls

Rockwell Collins Model Checking Example

- **Appendix C of the Formal Methods Supplement has an example of this in section FM.C.1.5.2***
- **Requirement:-**

> *If a DU is available, then it shall display some application.*

- **Stated formally as:-**

> AG(LEFT_DU_AVAILABLE -> LEFT_DU_APPLICATION != BLANK)
> AG(RIGHT_DU_AVAILABLE -> RIGHT_DU_APPLICATION != BLANK)
> *In all reachable states, if a DU is available, then its application shall not be blank.*

- **The model is analysed against these properties and if any counter examples can be found then they are flagged up.**
- **This method was used to analyse a model with 10^{120} states**

* Due to late format changes this reference may be incorrect

Aero Engine Controls

Airbus Unit Proof Example – LLR

- **Appendix C of the Formal Methods Supplement has an example of this in section FM.C.1.5.1***
- **Requirement:-**

The system shall
check the memory region between address A1F2_Memory_Zone and A1F2_Memory_Zone + A1F2_ZONE_SIZE - 1
If *all locations are set to the value 0xFF* ***then***
the system shall return OK.
Else
the system should return NOT_OK.

- **This is expressed as:-**

LET COND_FCT = ($\forall$ k $\in$ int. k>=0 $\wedge$ k<A1F2_ZONE_SIZE $\Rightarrow$
(A1F2_Memory_Zone.[.(k)]=0xFF));
The initial value is correct for all the indexes

LET COND_ERR = ($\exists$ k $\in$ int. k>=0 $\wedge$ k<A1F2_ZONE_SIZE $\wedge$
(A1F2_Memory_Zone.[.(k)] $\neq$ 0xFF));
There exists an index for which the initial value is wrong

* Due to late format changes this reference may be incorrect

Aero Engine Controls

Airbus Unit Proof Example – Code

```
T_RESULTAT A1F2_TestZone  ( )
{
    Rl_Return = OK ;
    Rl_Index = 0;
    while ((Rl_Index < A1F2_ZONE_SIZE) && (Rl_Return == OK))
    {
        if (A1F2_Memory_Zone[Rl_Index] != 0xFF)
        {
                Rl_Return = NOT_OK ;
        }
        Rl_Index = Rl_Index + 1 ;
    }
    return Rl_Return   ;
}
```

Problems with proof

- **Missing requirements**
  - **It is important to ensure that a complete set of requirements exists otherwise the proof of correctness is flawed.**
  - **It is possible to analyse requirements to ensure completeness. This is like adding requirements for all robustness tests.**
  - **If a component implements two functions with distinct I/O. Fully describing and proving one function leaves the second completely un-verified. Information flow analysis detects this.**

Using Information Flow Analysis to Detect Requirement Errors

- **Earlier we mentioned how Information flow plays a part in detecting missing requirements**
- **Lets take the previous example:-**

```
Derives A from B, C;
```

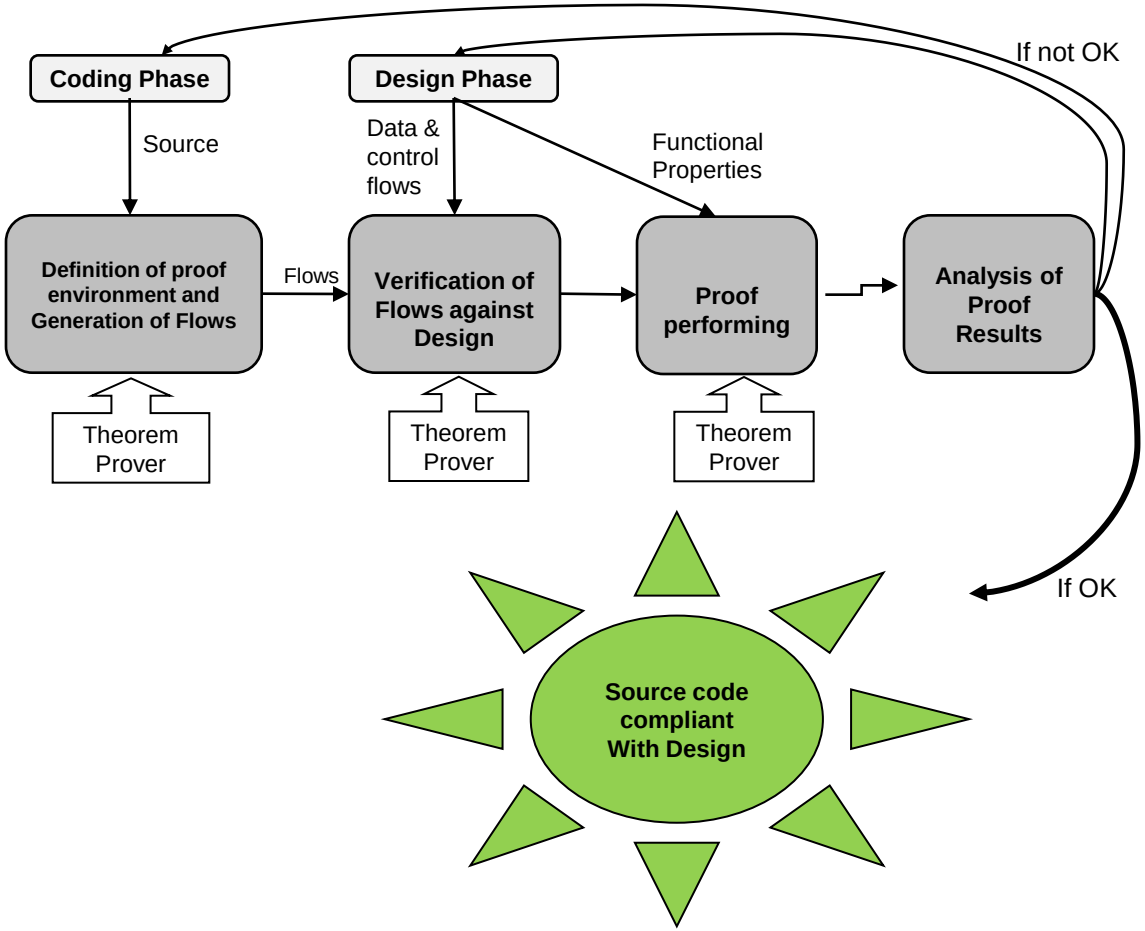
- **If we imagine the following code:-**

```
Begin
   A := B/C;
   D := B/F;
End;
```

- **It is very simple to prove that for all B and C the code produces the correct value of A.**
- **Information flow analysis would identify a missing requirement for deriving D from B and F.**

Aero Engine Controls

Process Overview for Unit Proof



Some Issues

- It must be ensured that the verification and any assumptions made for that verification are verified
  - This is harder with Formal Analysis
- Traceability between requirements and verification cases must be established.
- Analysis that all requirements have verification cases and that the sum of those cases fully verifies each requirement must be performed.
- Extraneous Code – Identification and removal or justification
  - With testing there are two concerns with Extraneous code.
    1. Due to bad design which needs to be addressed.
    2. A scenario where the functionality of the component is compromised
  - With proof it is established that there is no way in which the Extraneous code can compromise the functionality
  - However the existence of extraneous code may be an indication of bad design so it must still be identified and removed or justified.

Aero Engine Controls

Benefits

- **Formal Methods can:**
  - **improve the requirements by standardising them, ensuring some aspects of completeness, removing ambiguity**
  - **reduce the introduction of errors by improving requirements**
  - **improve error detection by allowing exhaustive, automated, mechanistic, repeatable verification on artifacts that could normally only be peer reviewed**
- **In a DO-178 life-cycle effort and time are proportional to:**
  - **the volume of errors**
  - **the point where errors are introduced and where they are detected**
- **The quality and safety of the product is dependent on the undetected errors and any misunderstandings of open problems**
- **The application of Formal Methods can:**
  - **prevent some error introduction**
  - **aid the early detection of some errors**
  - **provide compelling evidence of the absolute absence of some errors**

- **….BUT they will increase some aspect of the development costs**

Aero Engine Controls

Summary

- **A Formal Methods Supplement has been produced – and approved by plenary to move forward for acceptance; Acceptance should happen by RTCA in December and EUROCAE in January**
- **Objectives in the supplement can be used where Formal Methods are applied and DO-178C objectives where it is not**
- **This supplement is intended to facilitate an evolutionary change to software development processes allowing for partial/gradual adoption of formal methods**
- **The supplement contains a discussion paper that provides examples of the use of formal methods**

- **Our thanks to all the members of SG-6 that made this possible**

Philippe Baufreton (SAGEM)
Martin Beeby (Seaweed Systems)
Holger Blasum (Sysgo)
Matteo Bordin (AdaCore)
Duncan Brown (Aero Engine Controls)
Darren Cofer (Rockwell Collins)
Hervé Delseny (Airbus)
Vincent Dovydaitis (Foliage)
Louis Fabre (Eurocopter)
Kelly Hayhurst (NASA)
Gary Horan (FAA)
Ibrahim Habli (University of York)
Michael Hennell (LDRA)
Michael Holloway (NASA)
Jeff Joyce (Critical Systems Lab)
Emmanuel Ledinot (Dassault)
Frédéric Painchaud (DRDC-RDDC)
Cyrille Rosay (EASA)
Jamel Rauahi (Centre dEssais Aeronautique de Toulosue)
Martin Schwarz (TTTech)
Crystal Seguine (Airbus)
Jean Souyris (Airbus)
Elisabeth Strunk (Aerospace)
Nick Tudor (QinetiQ)
Rob Weaver (NATS)
Mike Whalen (Rockwell Collins)
Virginie Wiels (ONERA)

formerly, Peter Amey (Praxis)

Aero Engine Controls

A Rolls-Royce plc and Goodrich Corporation joint venture

Questions?

Duncan Brown

Aero Engine Controls

Duncan.Brown@aeroenginecontrols.com

People you can count on, products you can trust.