Formal Methods in Aerospace: Constraints, Assets and Challenges

Virginie Wiels – ONERA/DTIM

ONERA
THE FRENCH AEROSPACE LAB

return on innovation

Overview

1. Constraints
   certification
2. Assets
   industrial practice of formal methods
3. Challenges
   research themes at Onera

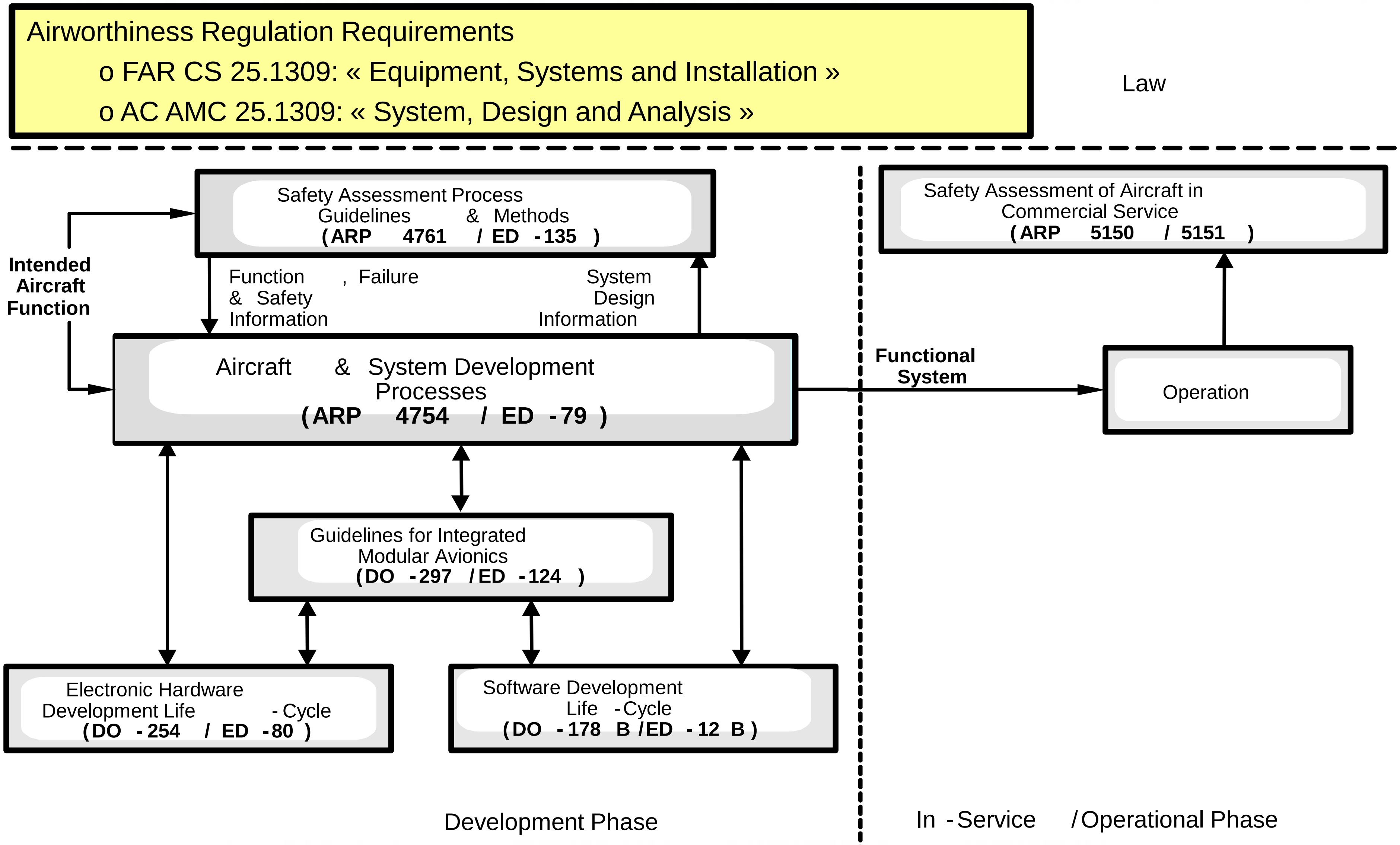
Focus on software

(but some information on systems, architectures and networks in 3)

Certification

- Negotiation between industrial company and certification authorities all along the development
  - EASA Europe
  - FAA USA
- For each aircraft
- Based on existing certification standards
- With negotiated specificities
  (Certification Review Item)

Aeronautic safety standards



Development Assurance Level

Relationships ARP 4754 / DO-178B

Software development assurance level is defined with respect to the criticality level of the system in which the software is included, to the potential consequences of the failure of this system

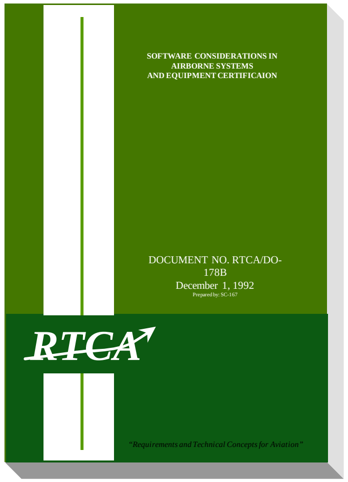
Certification objectives for software are then defined for each DAL by ED-12/DO-178.

Failure condition	DAL (development assurance level)
CAT (10^{-9})	A
HAZ (10^{-7})	B
MAJ (10^{-5})	C
MIN	D
No safety effect	E

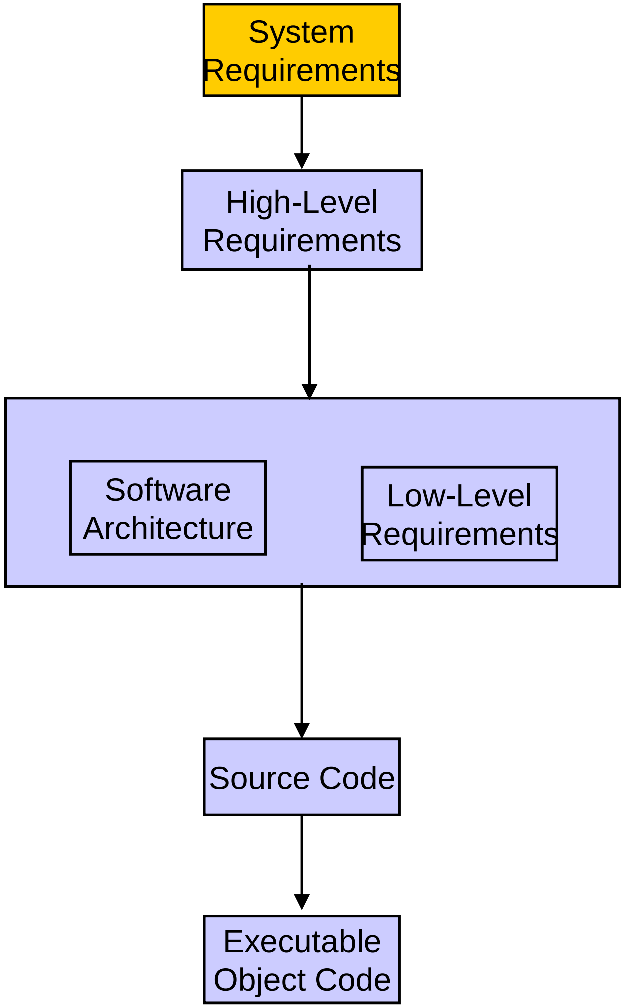
DO-178B

1. Introduction
2. System aspects relating to software development
3. **Software life cycle**
4. Software planning process
5. **Software development processes**
6. **Software verification process**
7. Software configuration management process
8. Software quality assurance process
9. Certification liaison process
10. Overview of aircraft and engine certification
11. Software life cycle data
12. Additional considerations

- **Annex A: Process objectives and outputs by software level**
- Annex B: Acronyms and glossary of terms Introduction



Software development processes



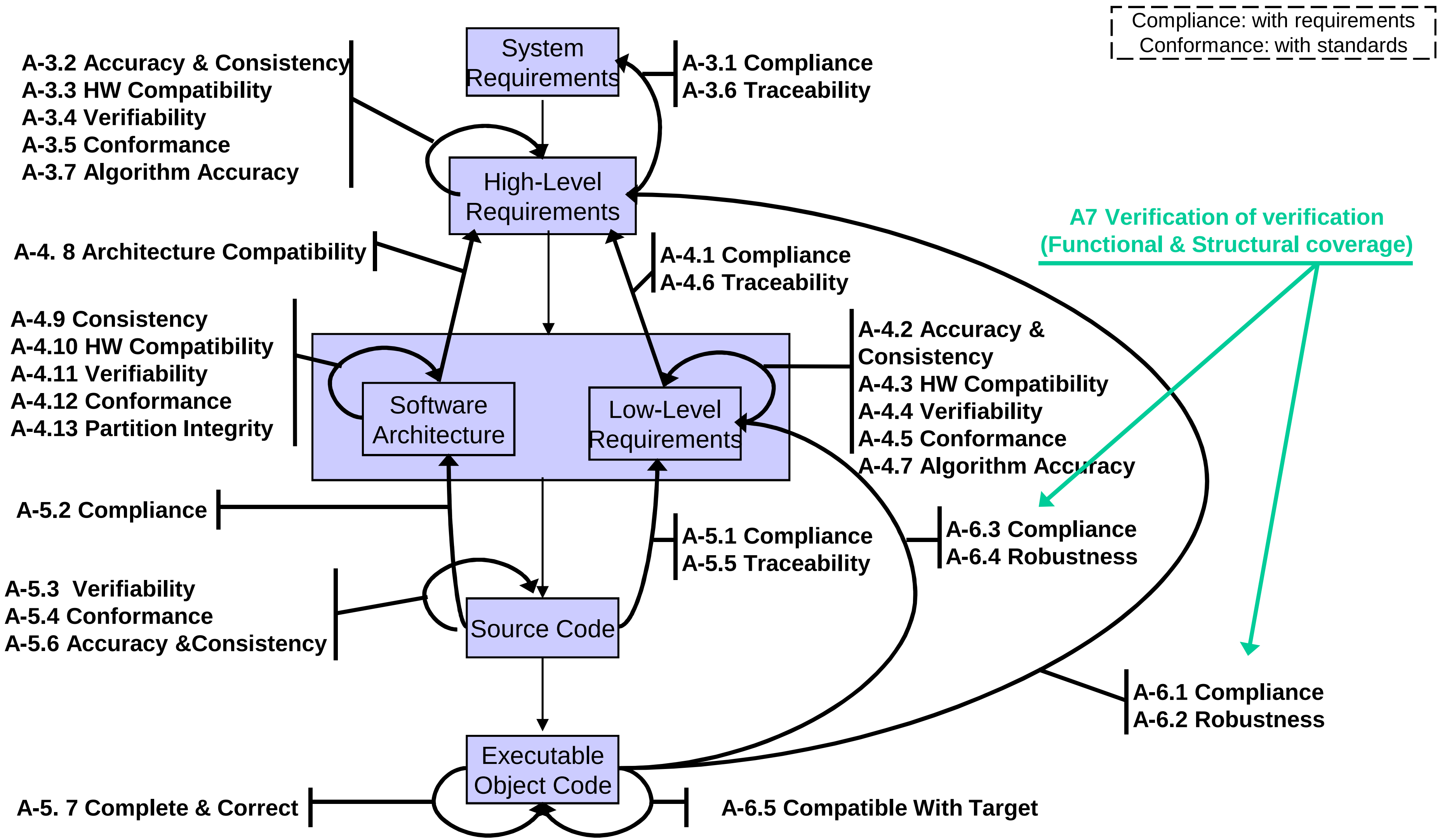
Software requirement process

Software design process

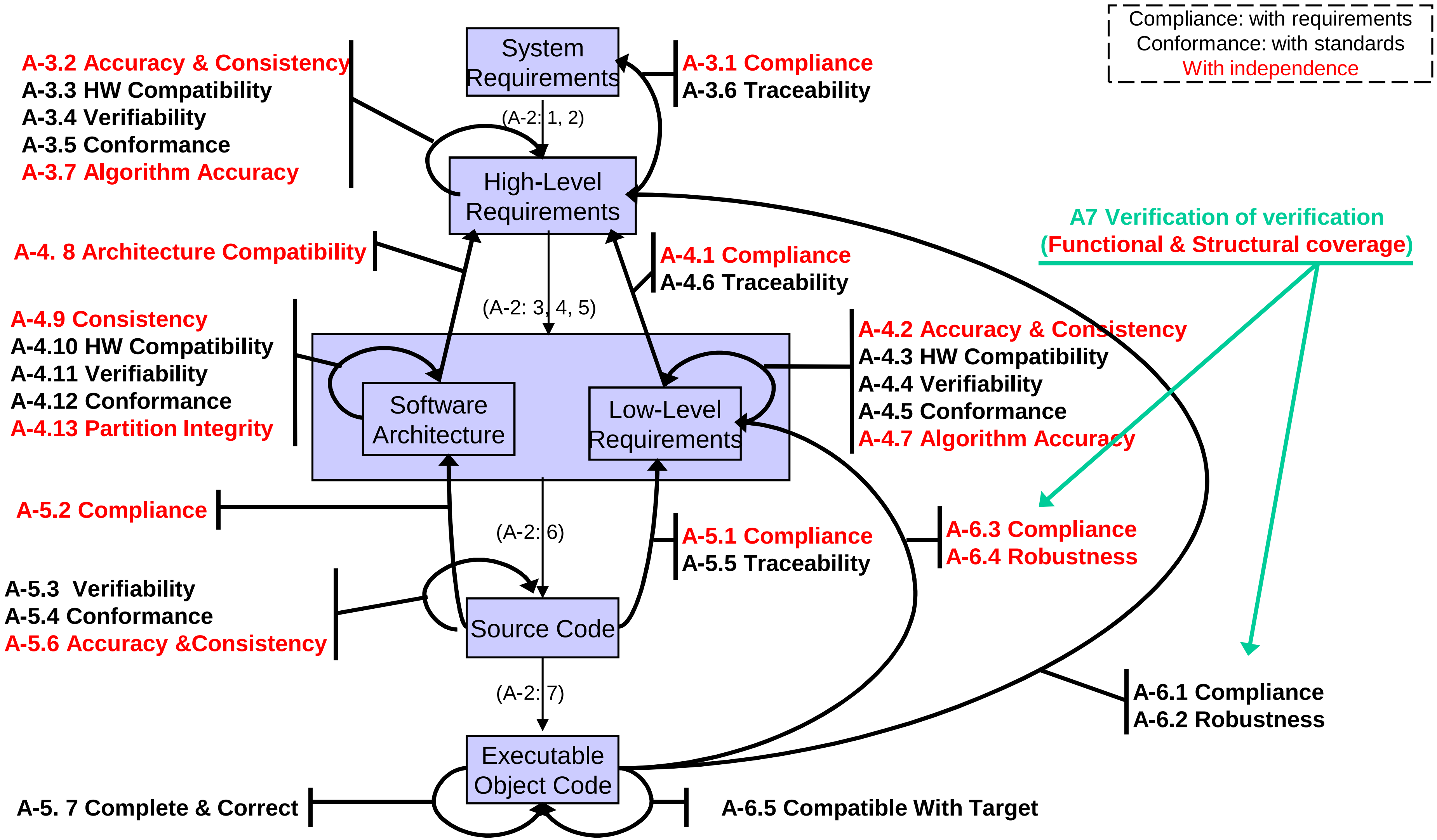
Software coding process

Software integration process

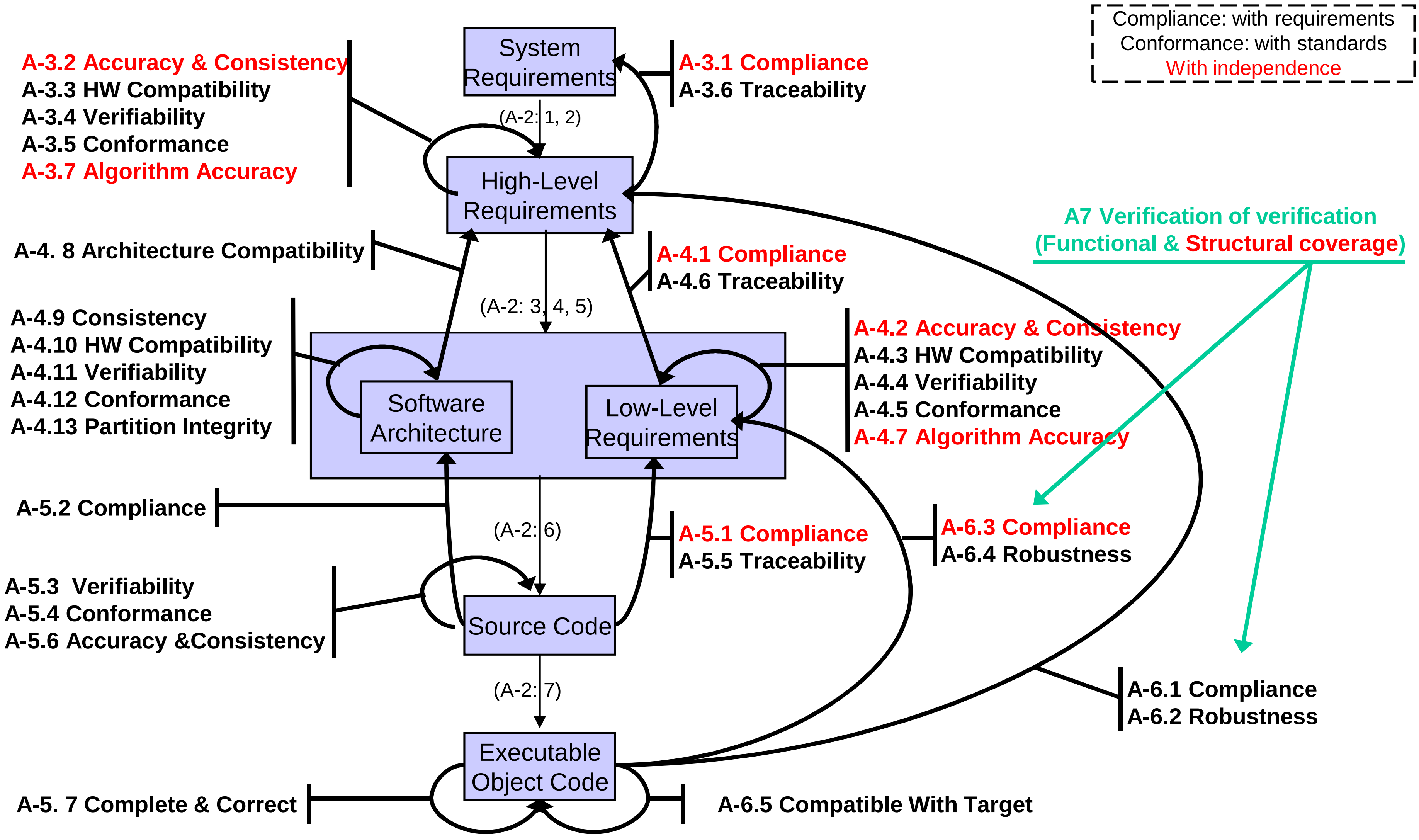
Software verification process objectives



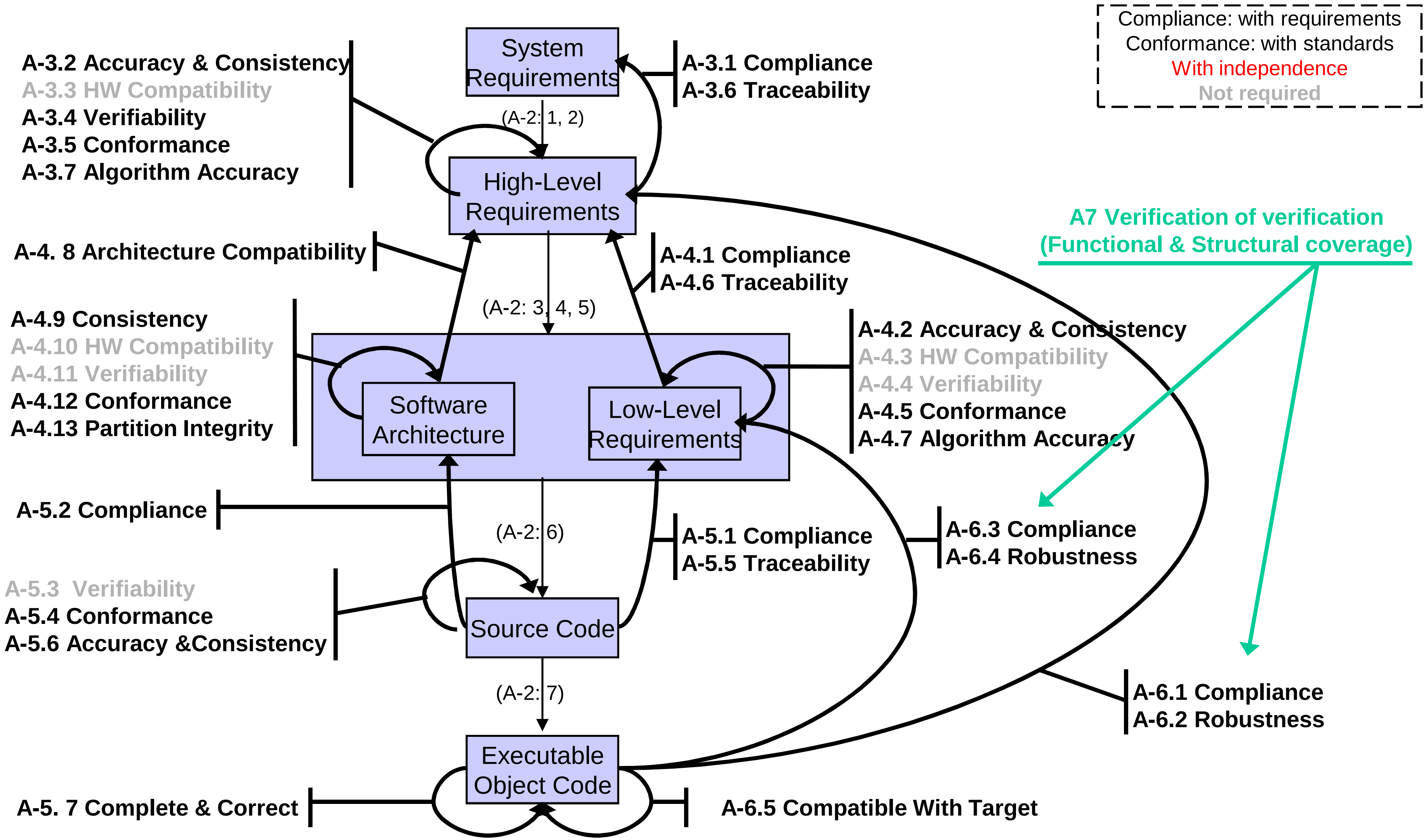
Verification process objectives level A



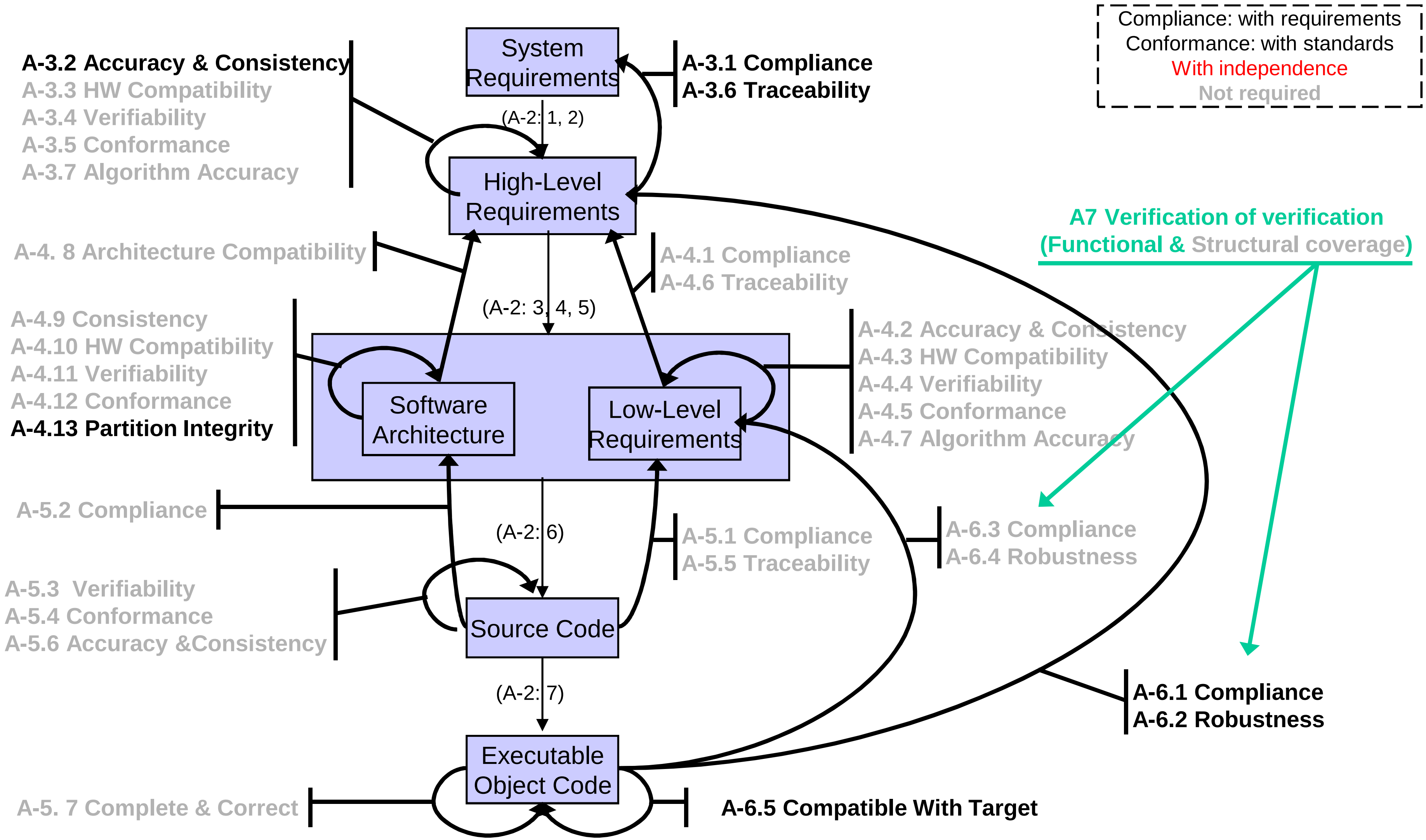
Software verification process : level B



Software verification process : level C



Software verification process : level D



Software verification process activities

- Reviews: qualitative assessment of correctness
- Analyses : repeatable assessment of correctness

  6.3 Software reviews and analyses

    6.3.1 Reviews and analyses of the HLR

    6.3.2 Reviews and analyses of the LLR

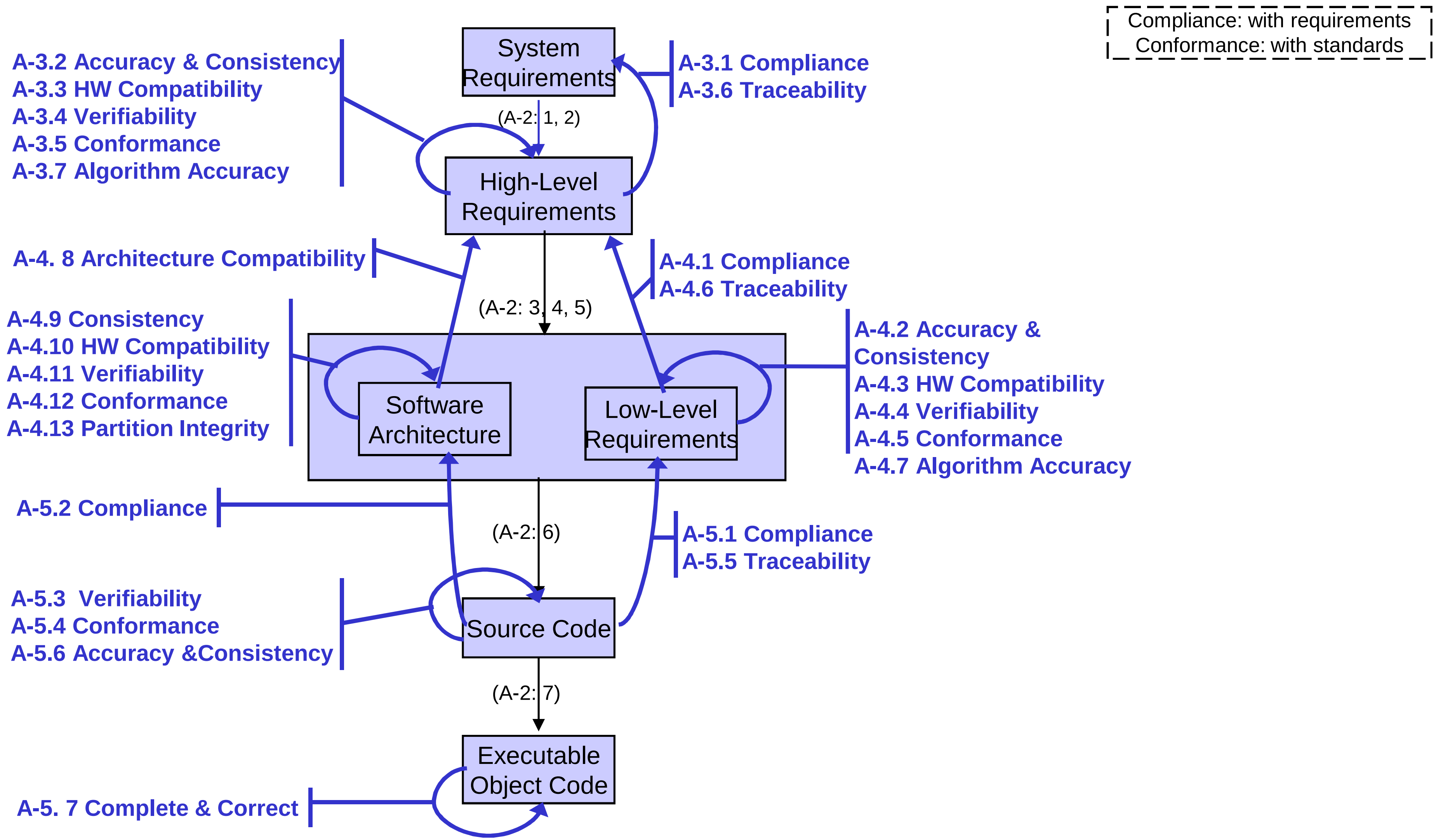
    6.3.3 Reviews and analyses of the software architecture

    6.3.4 Reviews and analyses of the source code

    6.3.5 Reviews and analyses of the outputs of the integration process

    6.3.6 Reviews and analyses of the test cases, procedures and results

Software verification process activities

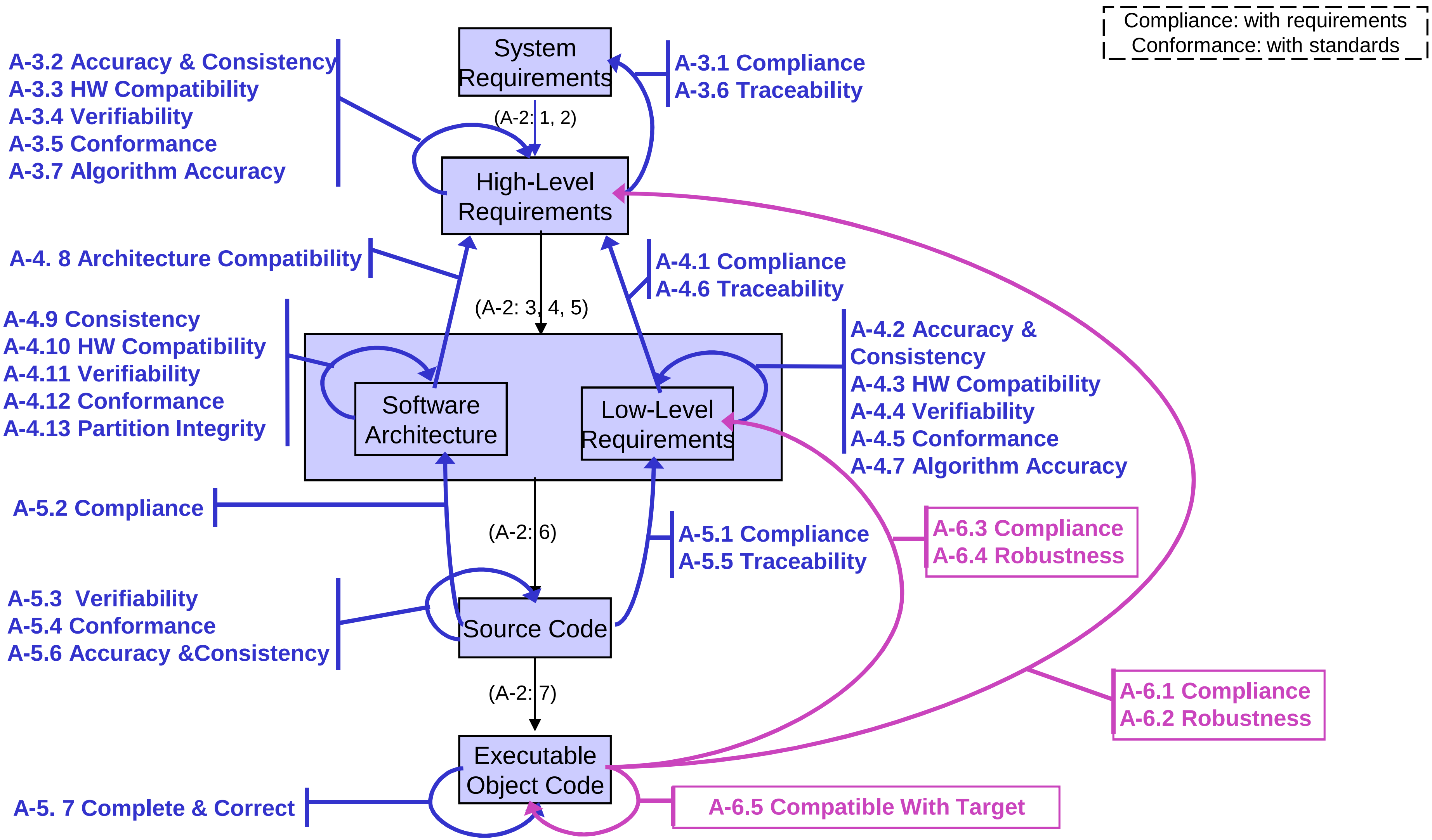


Software verification process activities

- Reviews: qualitative assessment of correctness
- Analyses : repeatable assessment of correctness

- Test
  - 6.4 Software testing process
    - 6.4.1 Test environment
    - 6.4.2 Requirements-based test case selection
    - 6.4.3 Requirements-based testing method
    - 6.4.4 Test coverage analysis

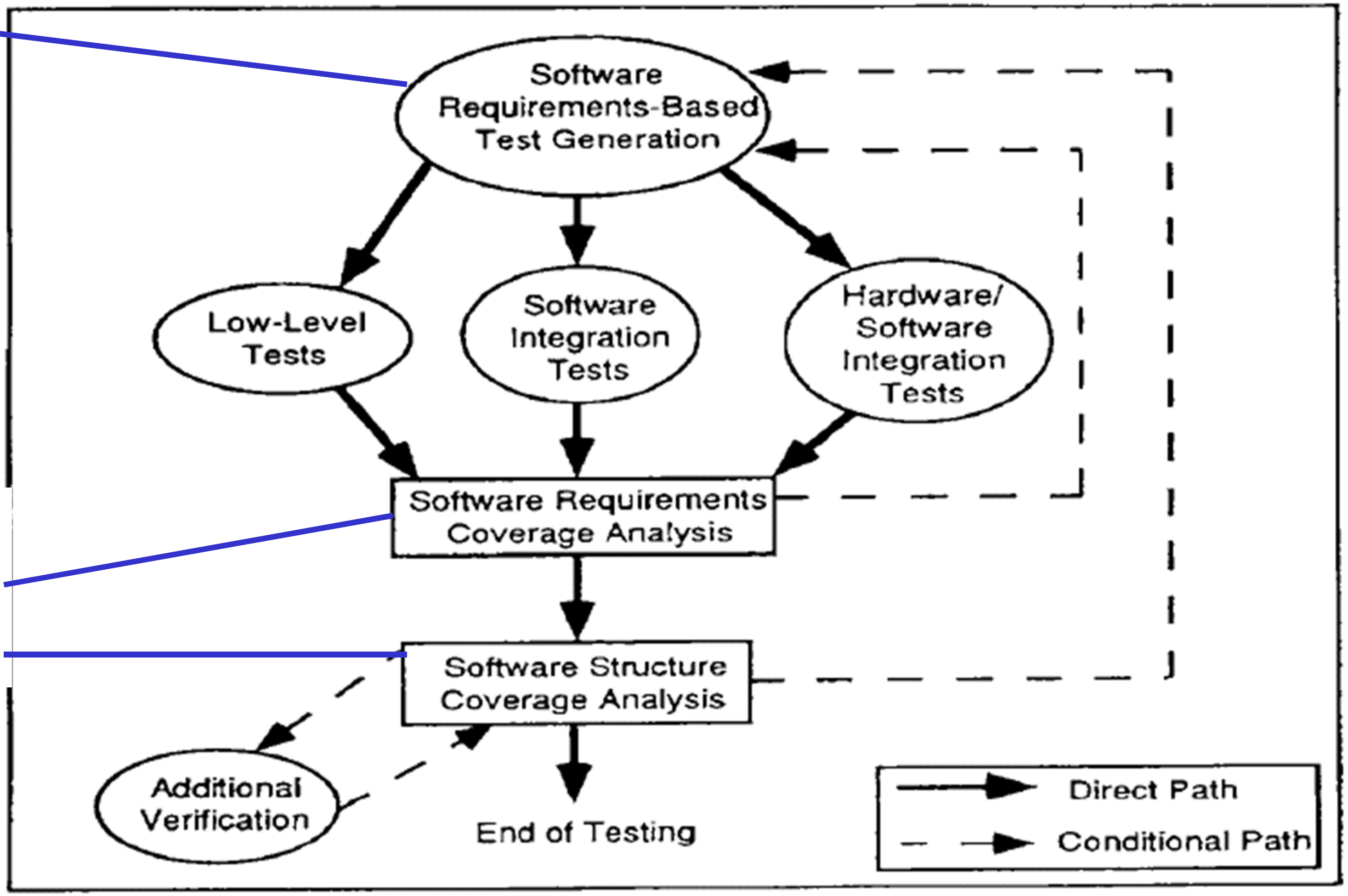
Software verification process activities



Test

Functional test only

Coverage analysis
- functional
- structural



Coverage

- Nominal and robustness test cases
- Functional coverage
  - At least one test case for each requirement (HLR and LLR)
- Structural coverage
  - Coverage criterai depending on DAL
    - MC/DC coverage level A
    - Decision coverage level B
    - Statement coverage level C
  - Dead code must be removed

DO-178C

- RTCA SC-205 / EUROCAE WG-71
  - 2005-2011
  - Industrials, certification authorities, tool vendors, experts
  - Consensus

- Outcome
  - Core document DO-178C
  - New document : DO-330 Tool qualification
  - Technical supplements
    - Model Based Development DO-331
    - Object-Oriented technologies DO-332
    - Formal Methods DO-333

DO-333: Formal Methods Technical Supplement

Enables the use of formal methods in replacement of traditional verification techniques

- Provides a guide for the use of formal methods
  - Modify existing objectives
  - Define new objectives
  - Describe activities
  - Define conditions for their use
- Provides information on formal methods
- Identifies and presents their characteristics

DO-333: Definition of Formal Methods

A model is an abstract representation of a given set of aspects of a system that is used for analysis, simulation, code generation, or any combination thereof.

A formal model is a model defined using a *formal notation*

Formal method — Formal model / Formal analysis

A *formal notation* is a notation having a precise, unambiguous, mathematically defined syntax and semantics.

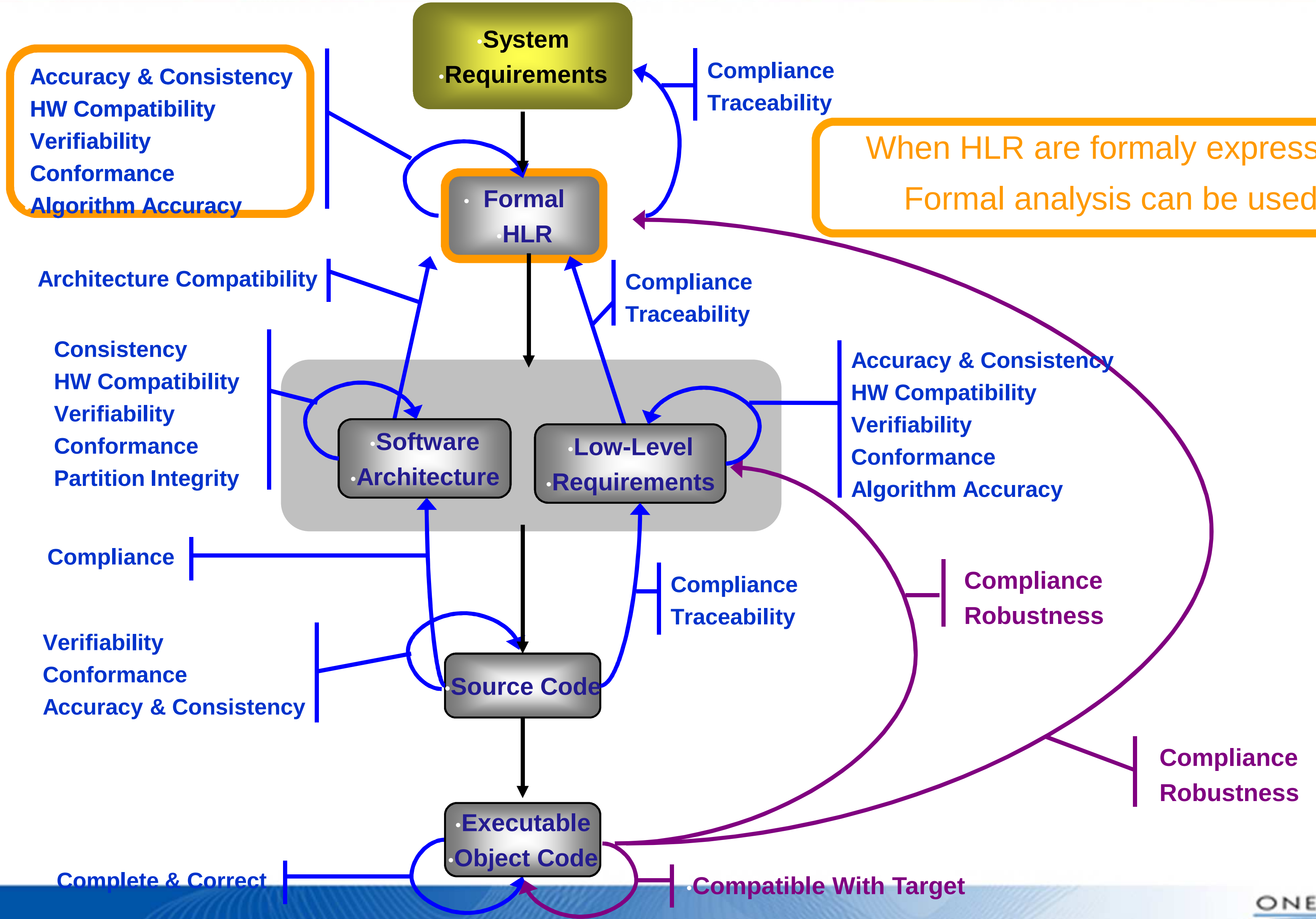
DO-333: Definition of formal methods

The use of mathematical reasoning to guarantee that properties are always satisfied by a *formal model.*

Formal method — Formal model, Formal analysis

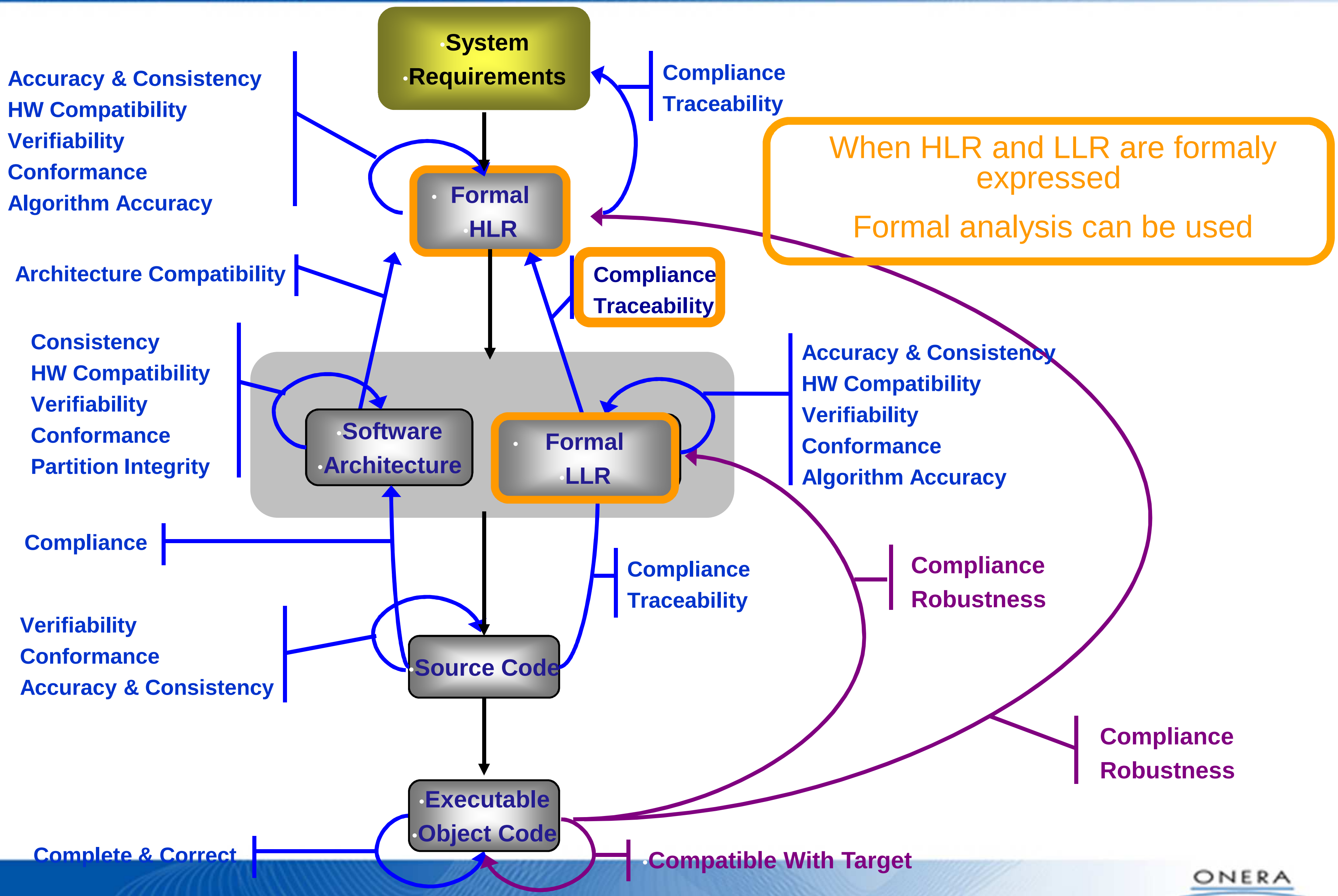
Soundness is required for an analysis to be considered formal

FM 6.3: software reviews and analyses

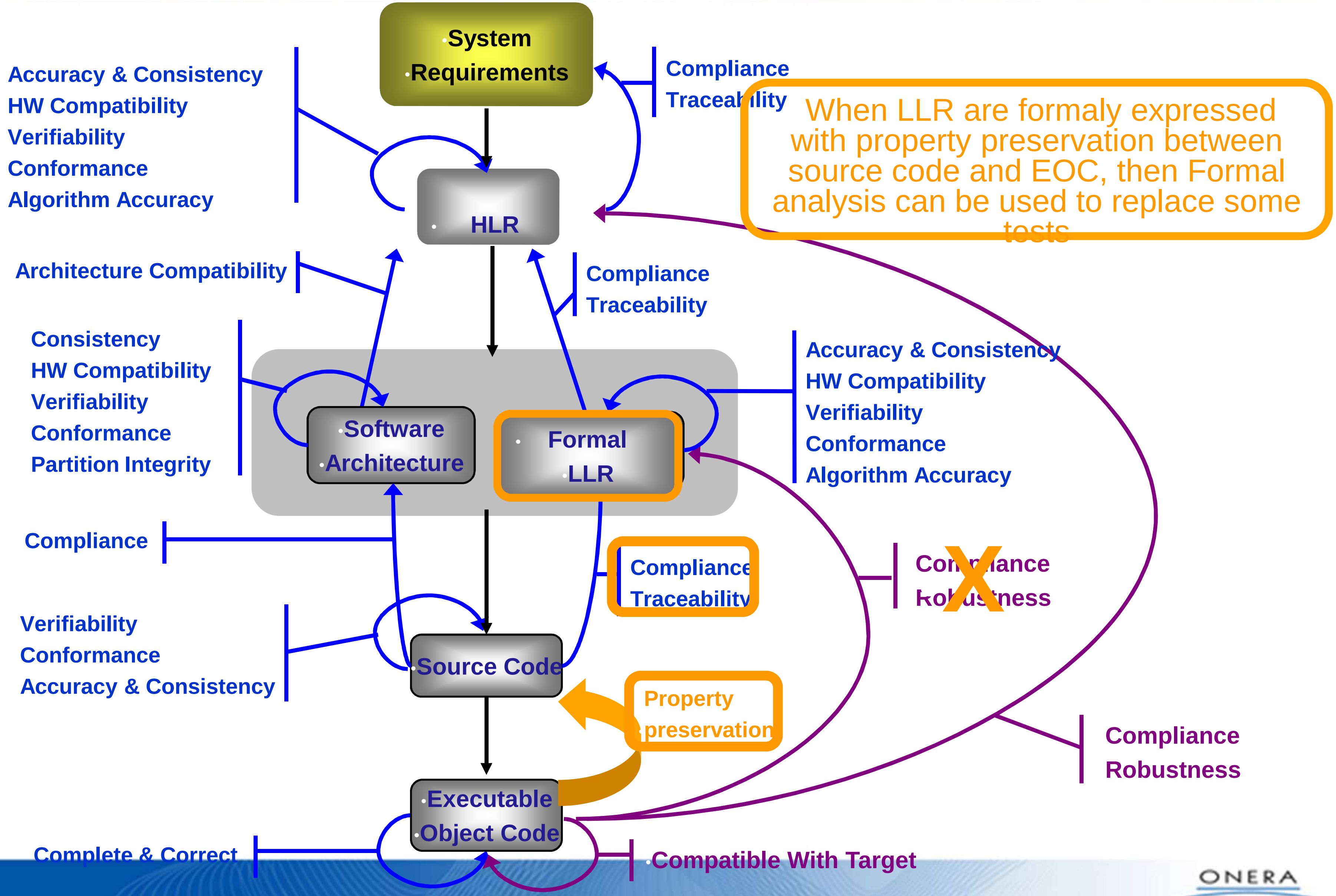


When HLR are formaly expressed
Formal analysis can be used

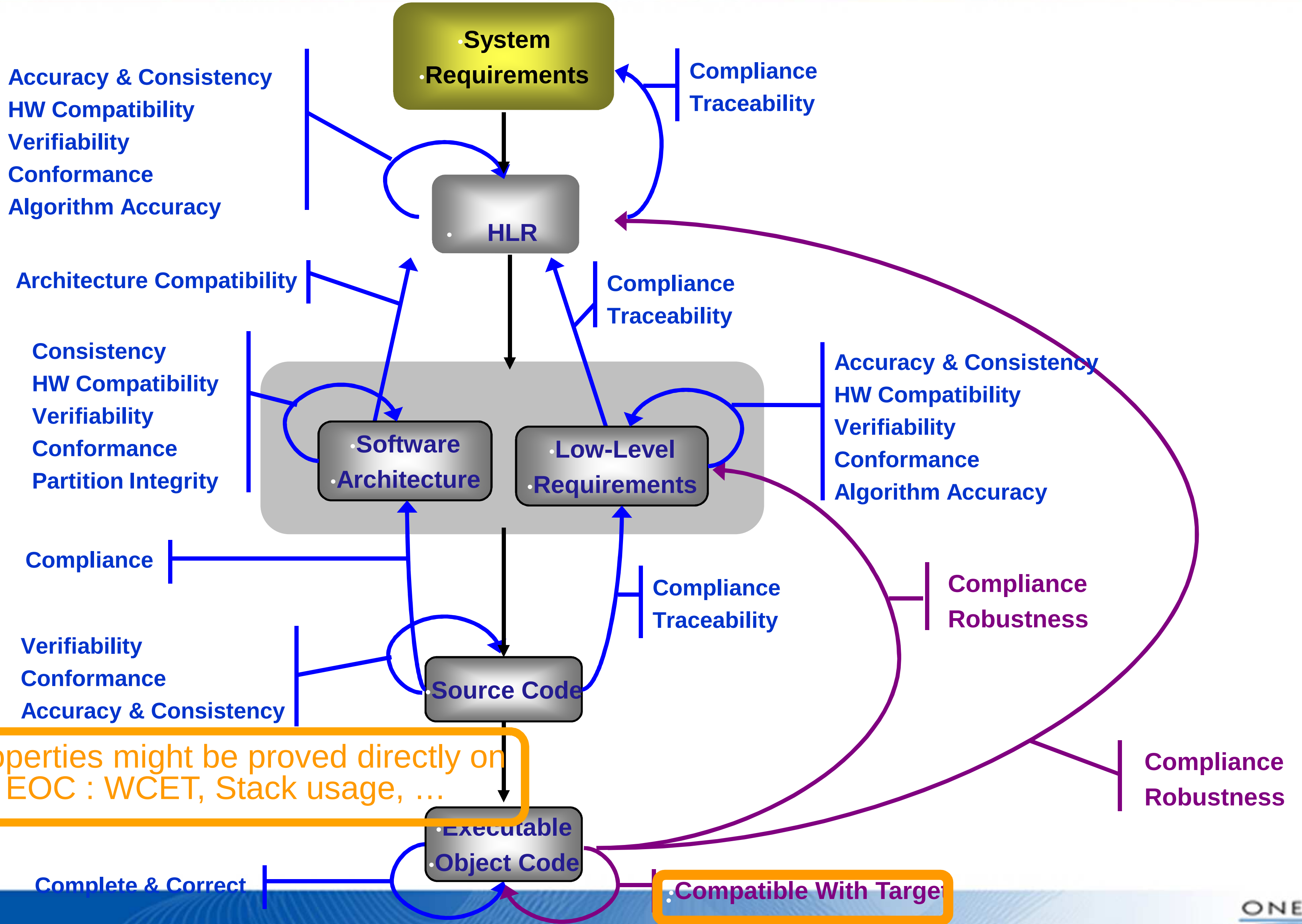
FM 6.3 : software reviews and analyses



FM 6.7 : Formal analyses of the EOC



FM 6.7 : Formal analyses of the EOC



Properties might be proved directly on EOC : WCET, Stack usage, …

FM Supplement : Formal verification

Formal Analysis might replace :

- Review and analysis objectives
- Conformance tests versus HLR & LLR
- Robustness tests

Formal Analysis might help for verification of compatibility with the hardware

Formal Analysis cannot replace HW/SW integration tests

Therefore testing will always be required.

FM 6.7.1 Principle of coverage analysis when using formal methods

- Test
  - Requirements-based coverage analysis
  - Structural coverage analysis

- Formal methods: the structural coverage objectives may be replaced by
  - Complete coverage of each requirement (6.7.1.2)
  - Completeness of the set of requirements (6.7.1.3)
  - Detection of unintended dataflow relationships (6.7.1.4)
  - Detection of extraneous code including dead code and deactivated code (6.7.1.5)

FM 6.7.1 Principle of coverage analysis when using formal methods

- Structural coverage analysis aims at detecting:
  - Shortcomings in requirements-based verification cases or procedures : 6.7.1.2
  - Inadequacies in software requirements : 6.7.1.3 + 6.7.1.4
  - Extraneous code, including dead code, and deactivated code : 6.7.1.5

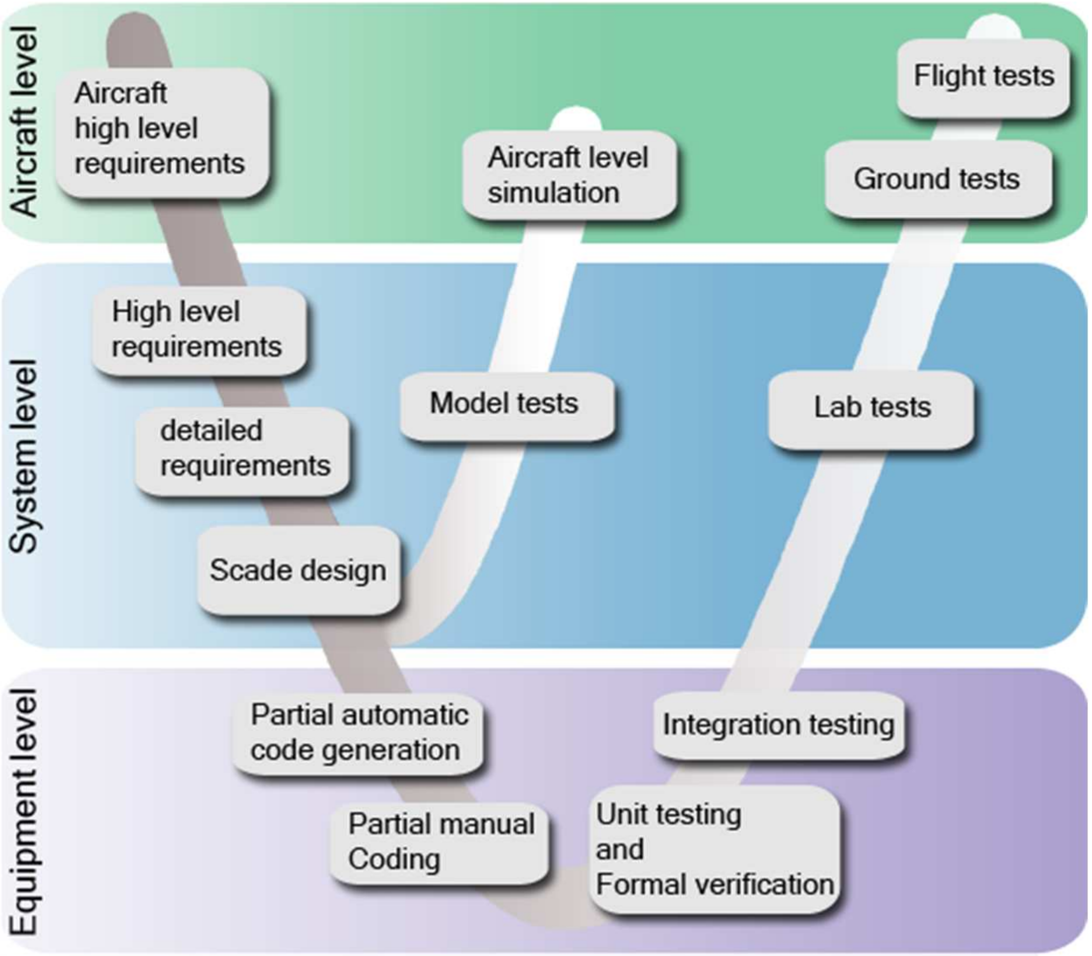
- Intuitively
  - FM ensure exhaustive coverage for a given requirement
  - To ensure complete coverage of the code, it remains to show that the set of requirements is complete wrt to the considered function

Overview

1. Constraints
   certification
2. Assets
   industrial practice of formal methods
3. Challenges
   research themes at Onera

Industrial practice: MBD

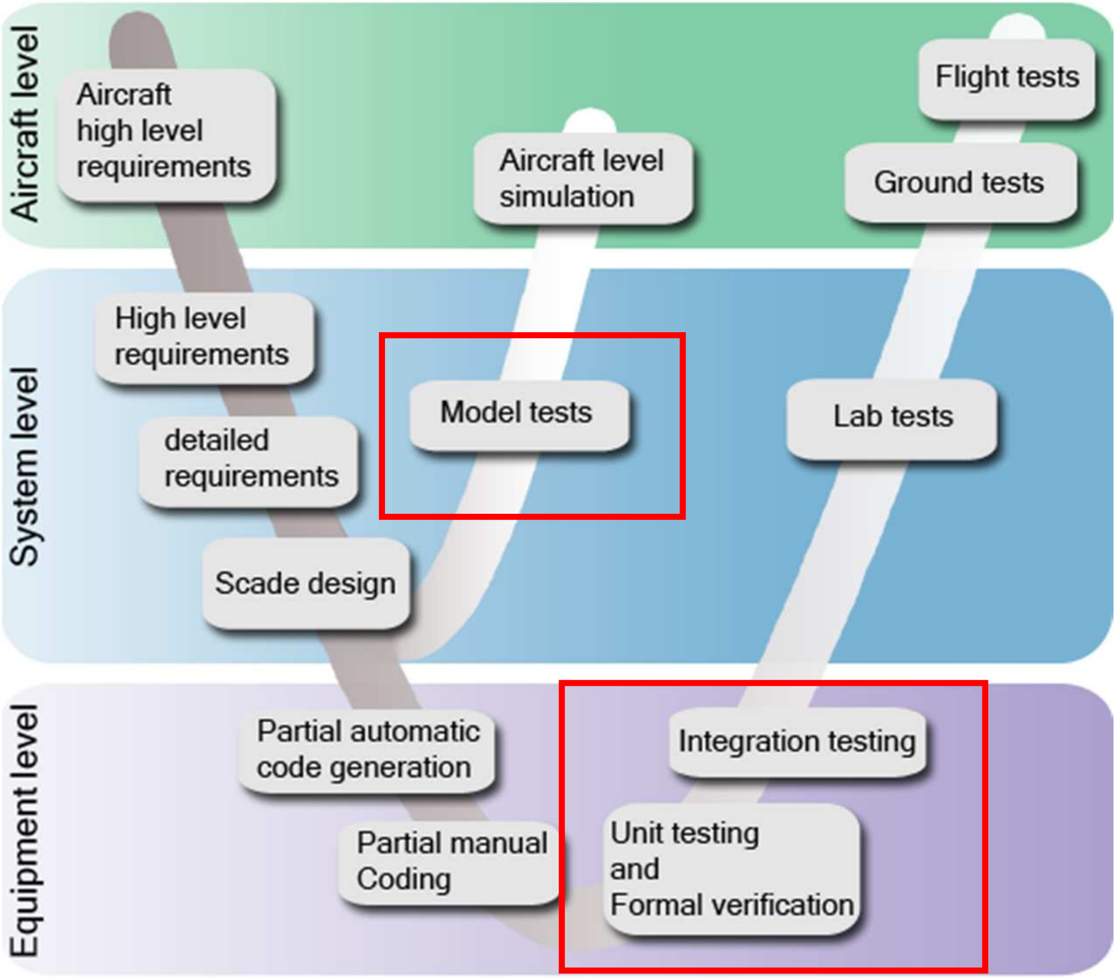
Model based development



Industrial practice: FM

- Models (Simulink, Scade)
  - Model checking
    - No certification credit yet
    - Better model earlier
- Source code (C, ada)
  - Proof of functional properties
    - DO-178 level A
- Model/code
  - Robustness analysis of models using static analysis on source code
- EOC
  - Abstract interpretation
    for stack analysis, wcet, absence of run-time errors
    - DO-178 level A, B, C

Airbus example



Experimenting model checking on Scade model

Absint
Frama-C
for DO-178 level A

Tools

- Frama-C frama-c.com
  - Extensible and collaborative platform
  - Dedicated to source-code analysis of C software
  - Connected to Z3, CVC3, Yices, Alt-Ergo, Coq, …
- Absint www.absint.com
  - Abstract interpretation based tools
  - Stack analysis
  - Wcet computation
  - Absence of run-time errors
- Tools have to be qualified (DO-330)

Industrial practice of formal methods

- 5 criteria defined by Airbus for the use of formal methods
  - Soundness
  - Cost Savings
  - Analysis of unaltered programs
  - Usability by normal software engineers on normal machines
  - Ability to be integrated into the DO-178B conforming process

A few references

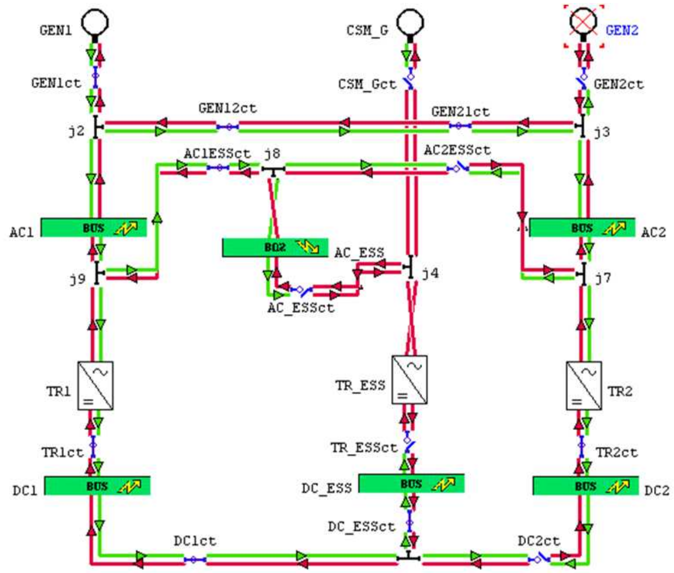
- *Testing or Formal Verification: DO-178C Alternatives and Industrial Experience*
  Yannick Moy, Emmanuel Ledinot, Hervé Delseny, Virginie Wiels, Benjamin Monate
  IEEE Software, 2013
- *Formal verification of avionics software products*
  Jean Souyris, Virginie Wiels, David Delmas, Hervé Delseny
  FM 2009
- *Model checking flight control systems: the Airbus experience*
  Thomas Bochot, Pierre Virelizier, Hélène Waeselynck and Virginie Wiels
  ICSE 2009
- www.onera.fr/staff/virginie-wiels

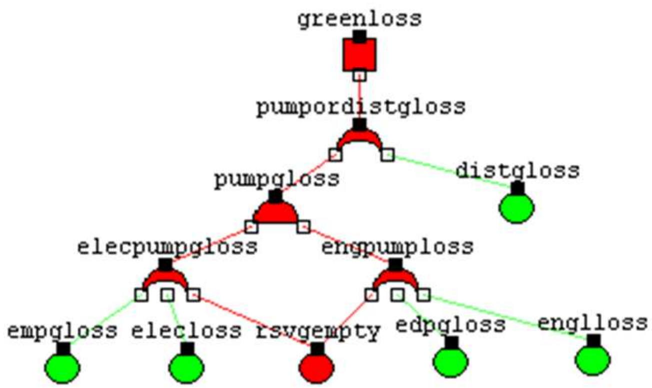
Overview

1. Constraints
   certification
2. Assets
   industrial practice of formal methods
3. Challenges
   research themes at Onera

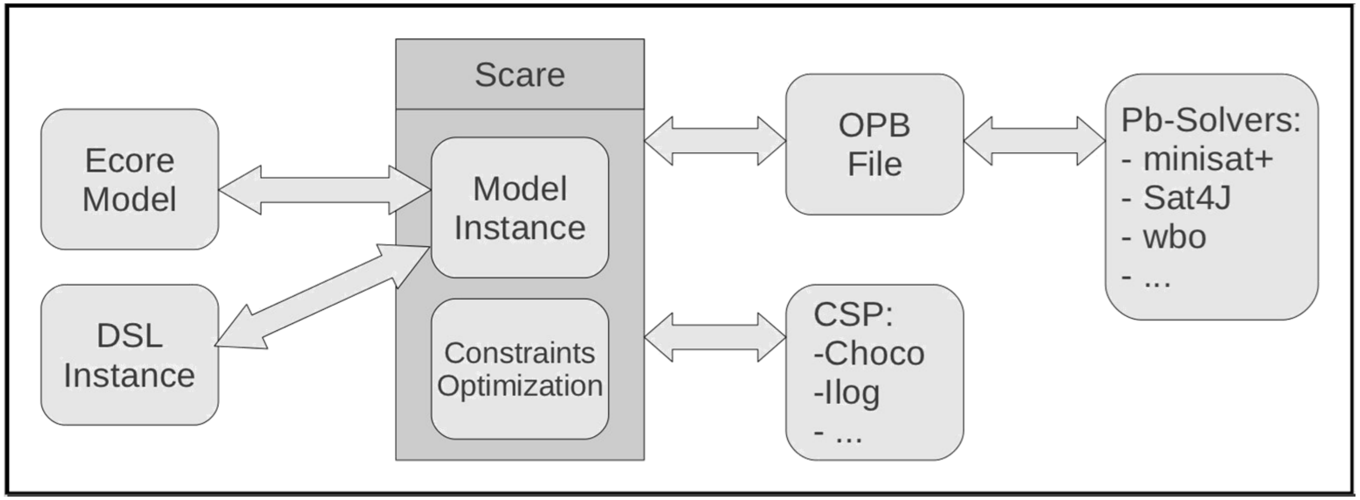
Formal safety assessment

- Formal models (Altarica)
- Evaluation
  - Elementary causes of a failure
  - Probability of failure
- Synthesis (solvers)
  - Independence relations
  - DAL allocation
    (Development Assurance Level)
- Industrial applications
  - Dassault (Falcon 7X)
  - Airbus
  - Astrium
- PoC: Pierre.Bieber@onera.fr





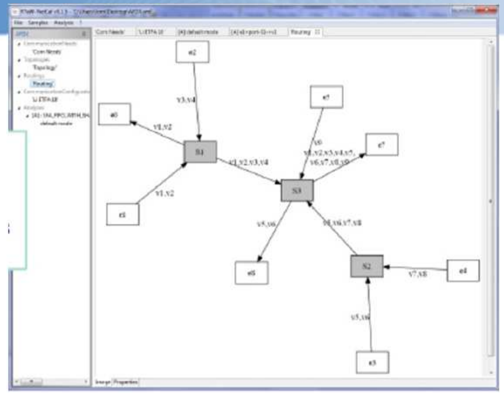
Architecture exploration

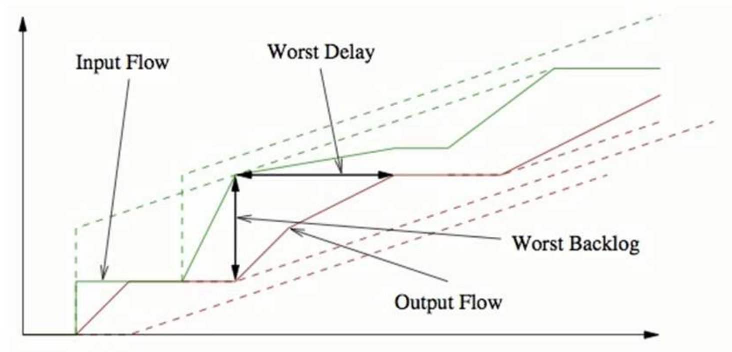


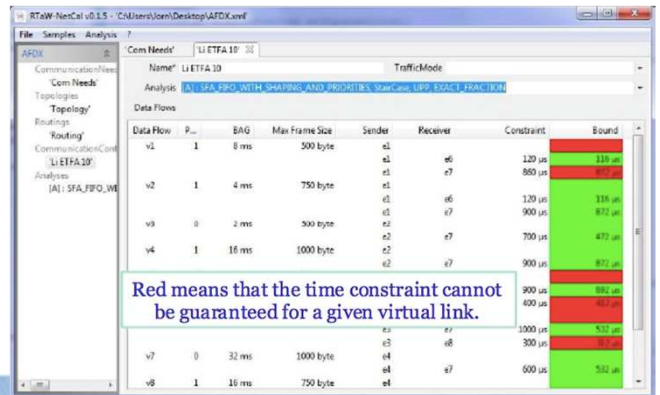
- Synthesis of correct solutions
  - From a set of constraints
  - Multi-viewpoints (Safety, Real Time, …)
- Design choices exploration/ dimensioning
  - Applied to allocation of functions on architectures
- PoC: David.Doose@onera.fr

Real Time assessment

- Worst Case Traversal Time
  - Commuted networks (AFDX…)
  - Network calculus
  - Tool developed with RTaW
  - PoC : Marc.Boyer @ onera.fr
- Worst Case Response Time
  - Includes functional level
  - Constraint solving
  - PoC : Frederic.Boniol@onera.fr
- Worst Case Execution Time
  - Probabilistic methods
  - PoC : Luca.Santinelli@onera.fr







Multi/Many-core architectures

Texas 8 cores

Freescale 8 cores

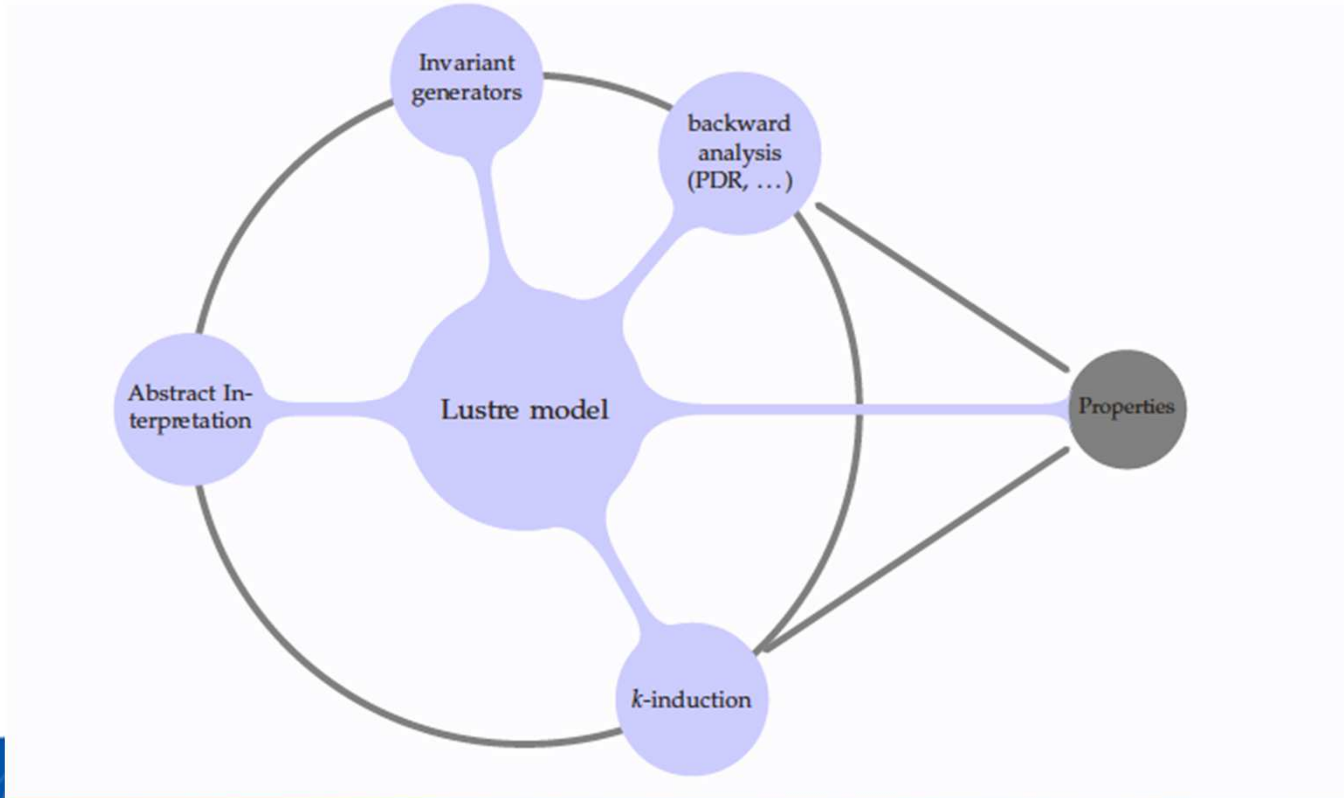
Tilera 32 cores

Kalray 256 cores

- Multi-many
  - Demonstration of determinism?
- Scheduling
  - Schedulability analysis
  - Off-line scheduling synthesis
- Code generation
  - Multi-threaded
- PoC: Eric.Noulard@onera.fr,Claire.Pagetti@onera.fr

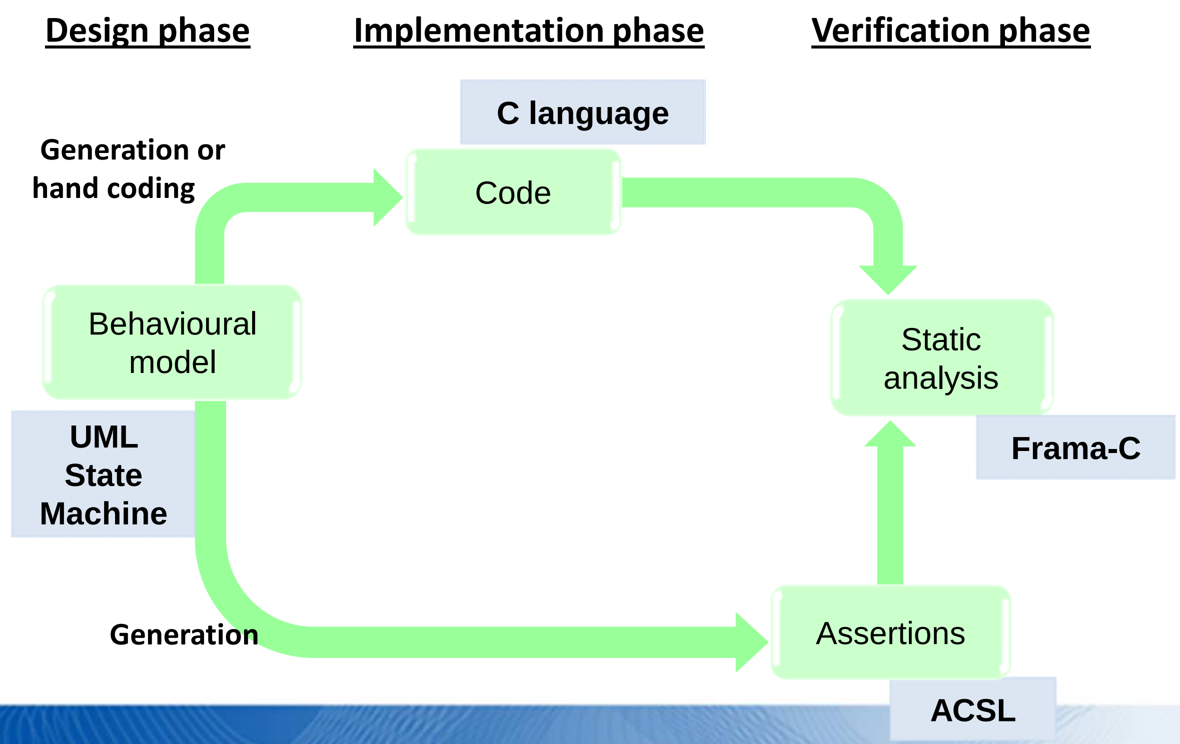
Cooperation of formal techniques

- Verification framework at model level (Lustre)
  - K-induction, backward analysis, invariant generation, AI
  - In collaboration with Rockwell-Collins
- Poc: Remi.Delmas@onera.fr



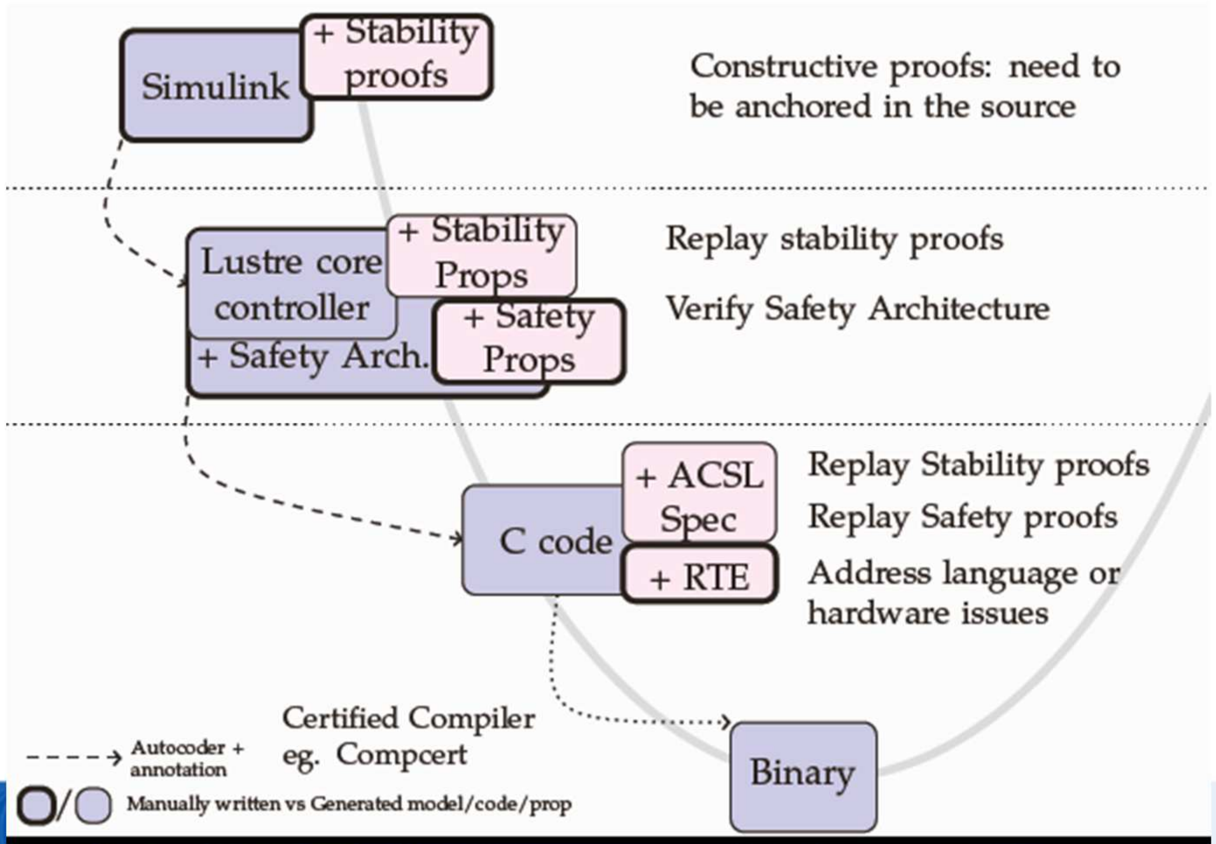
Software verification: model/code

- Formal proof of compliance of C code wrt UML state machine model (using Frama-C)
- PoC: Thomas.Polacsek@onera.fr



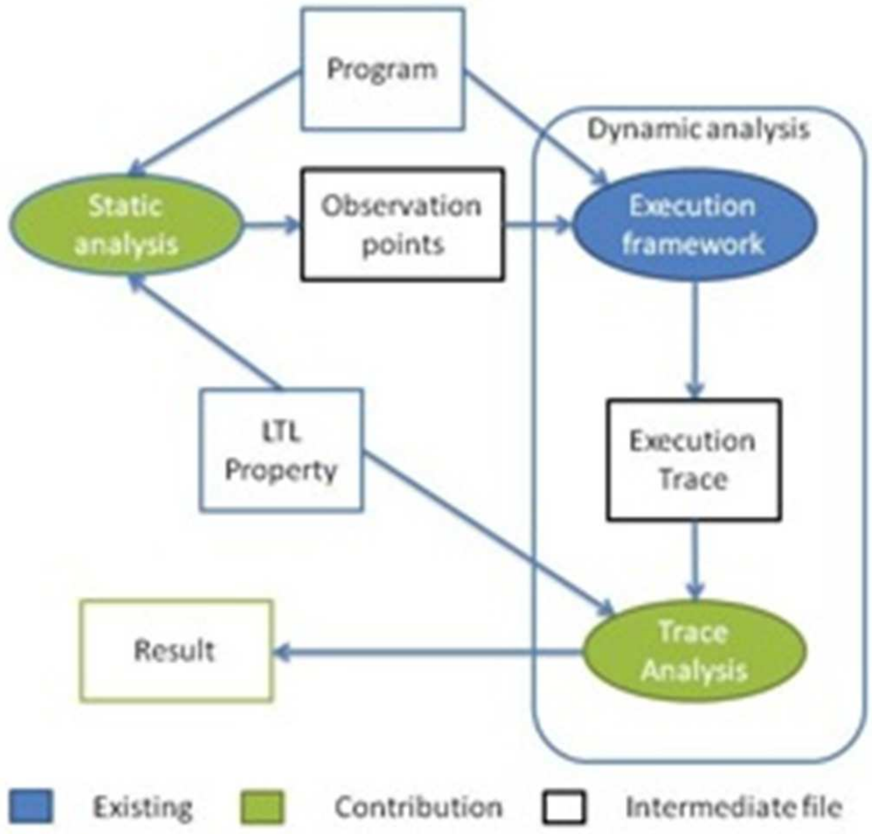
End-to-end verification of control-command systems

- Stability properties of control-command systems
  - Embedding properties all along the development
  - In collaboration with Georgia Tech, NASA, Iowa University
- PoC: Pierre-Loic.Garoche@onera.fr



Dynamic analysis and combination with test

- Formal verification of temporal properties on execution traces
  - Avionics software (Airbus)
  - Static analysis for the generation of observation points
  - Efficient verification (Büchi) for long traces
- Long-term objective
  - Finely combine static analysis, dynamic analysis and test



- PoC : Virginie.Wiels@onera.fr

Support to certification

- Software
  - Application of DO-333 (FM) and DO-331 (MBDV)
- Tools
  - Certification of FM tools
- IMA (Integrated Modular Architectures)
  - Support to certification authorities
  - Incremental certification
- ARP 4754
  - DAL allocation
- Multi/Manycore
  - Identification of specific issues for certification