Proposed Formal Methods Supplement for RTCA DO 178C

Dr. Jeffrey Joyce

Critical Systems Labs Inc.

jeff.joyce@cslabs.com

High Confidence Software and Systems 11th Annual Conference 3-6 May 2011, Annapolis

Abstract

This presentation describes key elements of a document prepared by RTCA SC 205 (Subgroup 6) that provides guidance on how formal methods may be used "for credit" in the certification of airborne software. This document is expected to become a supplement of RTCA DO 178C which, as the successor of RTCA DO 178B, will be used as the basis of certifying airborne software. In this presentation, Dr. Joyce will focus on what is expected from the use of formal methods to be "at least as good as" a conventional approach that does not use formal methods. For example, this presentation highlights guidance contained in this supplement concerning the use of formal methods to avoid unintended functionality, a.k.a. "undocumented code". Just as RTCA DO 178B has had a very wide influence over the past two decades on the development of software for high confidence systems beyond the avionics industry, it is likely that this supplement will serve as guidance for the use of formal methods in the development of many different kinds of high confidence system of interest to both the safety and security communities.

Key Ideas (of the FM supplement)

- The FM Supplement is only applicable if some formal analysis results are being used instead of some testing as part of the evidence submitted for certification
- Use of formal methods is an evolutionary change, not abrupt – as much or as little as you choose
- The supplement's provisions are very flexible in terms of "where" formal methods can be used in the life-cycle
- The FM supplement is very closely aligned with DO 178, e.g., structure of the supplements matches core document almost "paragraph by paragraph"
- Some testing is always necessary, even with the most comprehensive use of formal methods

Key Ideas (+1)

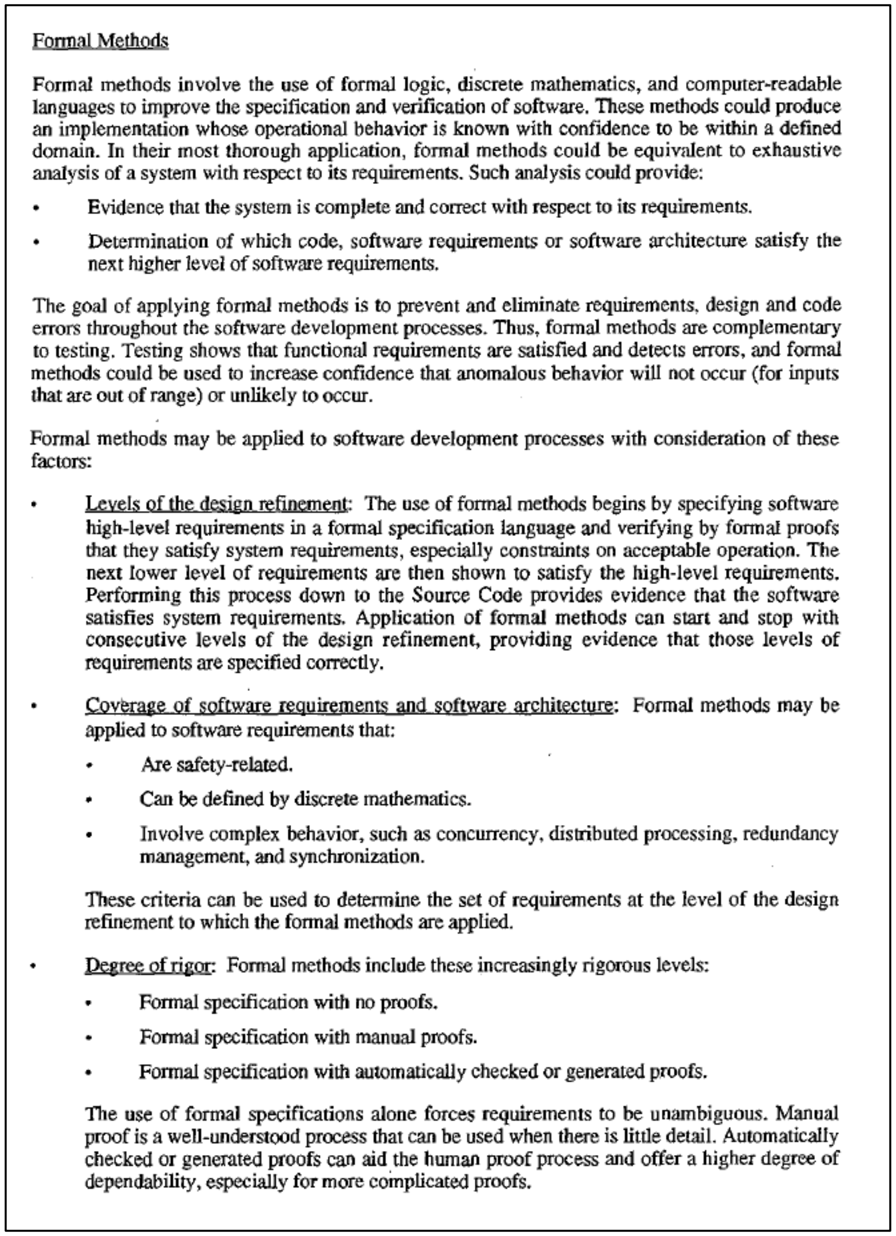
- Some of the more subtle concepts of the FM supplements are:
  - A formal model of a software development artifact must be a <u>conservative representation</u> of that artifact
  - Ensuring the <u>compliance and traceability objectives</u> of DO 178 includes showing that the outputs of software development step are necessary w.r.t. its inputs
  - If any testing is used to achieve compliance and traceability objectives for executable code, then <u>structural coverage</u> using a conventional test-based approach must be used

Background – RTCA DO 178B

- Used as a basis for the certification of airborne S/W
- Often used/consulted by other industries – very little of the content is unique to airborne S/W
- Is not a safety standard; instead, it is concerned with the correct and robust implementation of S/W
- Typically used in conjunction with SAE ARP 4761
- System requirements, including safety requirements, are input to the activities addressed by DO 178B
- 66 objectives at Level A (highest level of criticality)
- Addresses verification, but not validation
- Expected to be replaced by DO 178C in 2012 (?)

Background

- DO 178B, issued in December 1992, includes a reference to the possibility of using formal methods as an "alternate method"
- Onus entirely on the applicant to convince the certification authority that the alternate method can be used for credit towards certification



Formal Methods

Formal methods involve the use of formal logic, discrete mathematics, and computer-readable languages to improve the specification and verification of software. These methods could produce an implementation whose operational behavior is known with confidence to be within a defined domain. In their most thorough application, formal methods could be equivalent to exhaustive analysis of a system with respect to its requirements. Such analysis could provide:

- Evidence that the system is complete and correct with respect to its requirements.
- Determination of which code, software requirements or software architecture satisfy the next higher level of software requirements.

The goal of applying formal methods is to prevent and eliminate requirements, design and code errors throughout the software development processes. Thus, formal methods are complementary to testing. Testing shows that functional requirements are satisfied and detects errors, and formal methods could be used to increase confidence that anomalous behavior will not occur (for inputs that are out of range) or unlikely to occur.

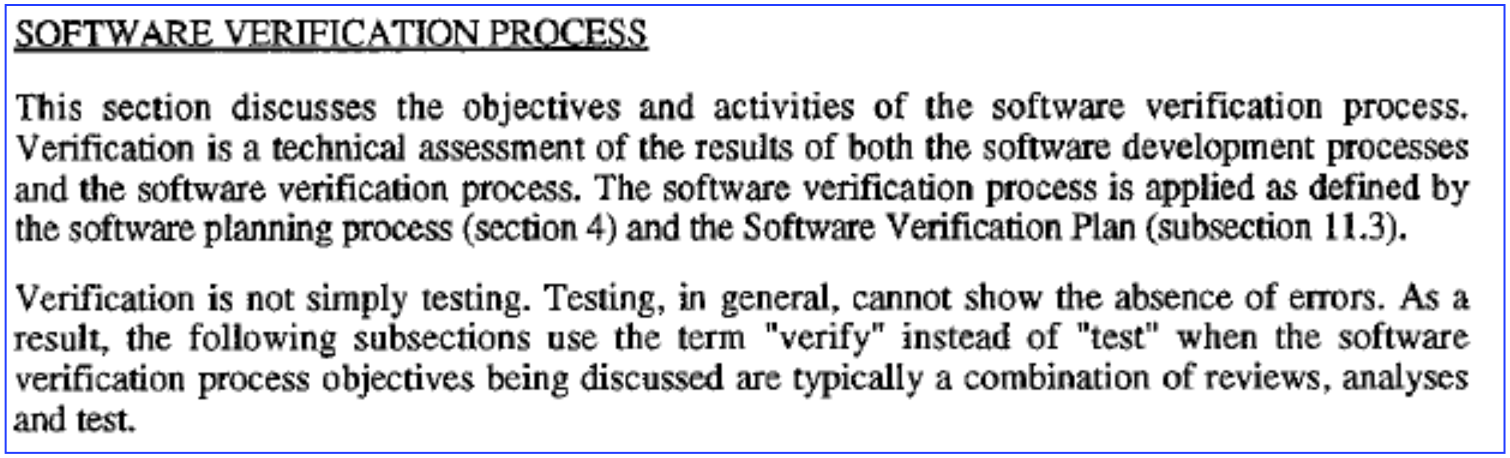
Formal methods may be applied to software development processes with consideration of these factors:

- Levels of the design refinement: The use of formal methods begins by specifying software high-level requirements in a formal specification language and verifying by formal proofs that they satisfy system requirements, especially constraints on acceptable operation. The next lower level of requirements are then shown to satisfy the high-level requirements. Performing this process down to the Source Code provides evidence that the software satisfies system requirements. Application of formal methods can start and stop with consecutive levels of the design refinement, providing evidence that those levels of requirements are specified correctly.
- Coverage of software requirements and software architecture: Formal methods may be applied to software requirements that:
  - Are safety-related.
  - Can be defined by discrete mathematics.
  - Involve complex behavior, such as concurrency, distributed processing, redundancy management, and synchronization.

  These criteria can be used to determine the set of requirements at the level of the design refinement to which the formal methods are applied.
- Degree of rigor: Formal methods include these increasingly rigorous levels:
  - Formal specification with no proofs.
  - Formal specification with manual proofs.
  - Formal specification with automatically checked or generated proofs.

  The use of formal specifications alone forces requirements to be unambiguous. Manual proof is a well-understood process that can be used when there is little detail. Automatically checked or generated proofs can aid the human proof process and offer a higher degree of dependability, especially for more complicated proofs.

Background (+1)


SOFTWARE VERIFICATION PROCESS

This section discusses the objectives and activities of the software verification process. Verification is a technical assessment of the results of both the software development processes and the software verification process. The software verification process is applied as defined by the software planning process (section 4) and the Software Verification Plan (subsection 11.3).

Verification is not simply testing. Testing, in general, cannot show the absence of errors. As a result, the following subsections use the term "verify" instead of "test" when the software verification process objectives being discussed are typically a combination of reviews, analyses and test.

- While the software verification section of DO 178B recognizes that verification is more than just "testing", much of the guidance and many of the corresponding objectives very explicitly refer to the use of test

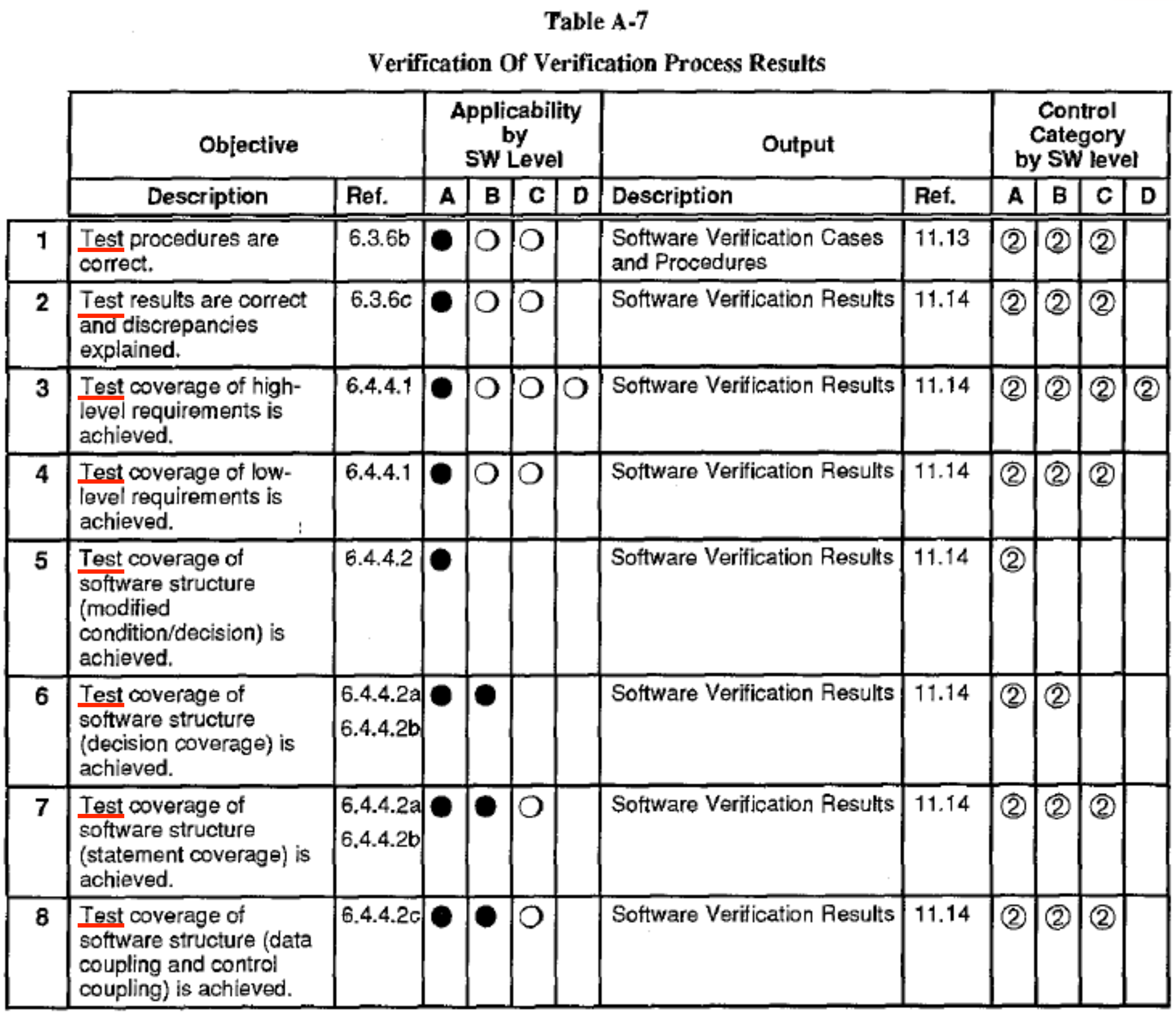


Table A-7

Verification Of Verification Process Results

	Objective		Applicability by SW Level				Output		Control Category by SW level			
	Description	Ref.	A	B	C	D	Description	Ref.	A	B	C	D
1	Test procedures are correct.	6.3.6b	●	○	○		Software Verification Cases and Procedures	11.13	②	②	②	
2	Test results are correct and discrepancies explained.	6.3.6c	●	○	○		Software Verification Results	11.14	②	②	②	
3	Test coverage of high-level requirements is achieved.	6.4.4.1	●	○	○	○	Software Verification Results	11.14	②	②	②	②
4	Test coverage of low-level requirements is achieved.	6.4.4.1	●	○	○		Software Verification Results	11.14	②	②	②	
5	Test coverage of software structure (modified condition/decision) is achieved.	6.4.4.2	●				Software Verification Results	11.14	②			
6	Test coverage of software structure (decision coverage) is achieved.	6.4.4.2a 6.4.4.2b	●	●			Software Verification Results	11.14	②	②		
7	Test coverage of software structure (statement coverage) is achieved.	6.4.4.2a 6.4.4.2b	●	●	○		Software Verification Results	11.14	②	②	②	
8	Test coverage of software structure (data coupling and control coupling) is achieved.	6.4.4.2c	●	●	○		Software Verification Results	11.14	②	②	②	

Background (+3)

- Users of DO 178B submitted “issues” to RTCA and Eurocae regarding the difficulty of getting credit for the use of formal methods
- For example ….

“There is no objective criteria for evaluating a formal methods project. The process for getting a formal proof through the certification process is not well documented. European documents (such as IEC 61508) recognizes formal methods/ proofs. Advances have been made in the area of formal methods that are now more practical and viable in "real life" than when DO-178B was written. Do not want to "recommend" the use of formal languages/methods but do need to know how to address it if it does arise on projects.”

Airbus 380

- Formal methods were used for certification credit in development of the A380, but apparently it was not a trivial matter to persuade certification authorities that this was acceptable even with the reference to formal methods in DO 178B as an "alternative method"

Background (+4)

- In addition to the Airbus experience, other factors were likely motivations for a decision in 2005 to dedicate a subgroup to the task of addressing formal methods …
  - An increasing discomfort about the limitations of software testing, especially with increasing complexity of airborne S/W due to technological advances such as IMA
  - R&D investment in formal methods by airborne S/W suppliers and tool vendors
  - Recognition of formal methods by comparable guidance and standards used in other industry sectors, e.g., IEC 61508
  - Successful and beneficial use of formal methods in other industry sectors
  - Anticipated impact of verification process of other "technologies" such as object orientation and model based development

Critical Systems Labs
Strategic Insight for Safety

Sub-Group 6 Membership

Sub - Group 6 Chairmen

EUROCAE WG-71:	Duncan Brown	Aero Engine Controls
RTCA SC-205:	Kelly Hayhurst	NASA

Sub-Group 6 Membership

Peter Amey	Praxis Critical Systems
Philippe Baufreton	SAGEM
Martin Beeby	Seaweed Systems
Matteo Bordin	Adacore Technologies
Darren Cofer	Rockwell Collins
Hervé Delseny	Airbus
Vincent Dovydaitis	Foliage
Louis Fabre	Eurocopter
Frederic Painchaud	Defence Research and Development Canada
Ibrahim Habli	University of York
Michael Hennell	LDRA
Michael Holloway	NASA
Gary Horan	FAA
Jeffrey Joyce	Critical Systems Labs
Emmanuel Ledinot	Dassault Aviation
Cyrille Rosay	EASA
Jamel Rouahi	CEAT
Martin Schwarz	TT Technologies
Jean Souyris	Airbus
Elisabeth Strunk	Aerospace Corporation
Nick Tudor	Qinetiq
Mike Whalen	Rockwell Collins
Virginie Wiels	Onera

… as it was during some point in the formative phase of this activity, approx. 2006 to 2009

Background (+5)

- First meeting of SC 205/WG 71 held March 2005
- Subgroup 6 (SG6) was formed to address a list of formal methods issues collected by RTCA and Eurocae
- Two main categories of issues …
  - Obstacles in DO 178B to using formal methods towards certification credit, e.g., "test" specific objectives
  - Difficulties, concerns and questions about meeting verification objectives for certain aspects of complex S/W where formal methods might be helpful

Five years later …

- The formal methods supplement for DO 178B by approved by the RTCA SC 250 plenary in 2010
- It is expected to be one of four "technology supplements" to be bundled with DO 178C
- This supplement mirrors the structure of DO 178B
  - Uses the language and terminology of DO 178B even though some of this might look odd to the formal methods community
  - Only shows what is, or could be, different when using formal methods

Section 6.0 - Verification

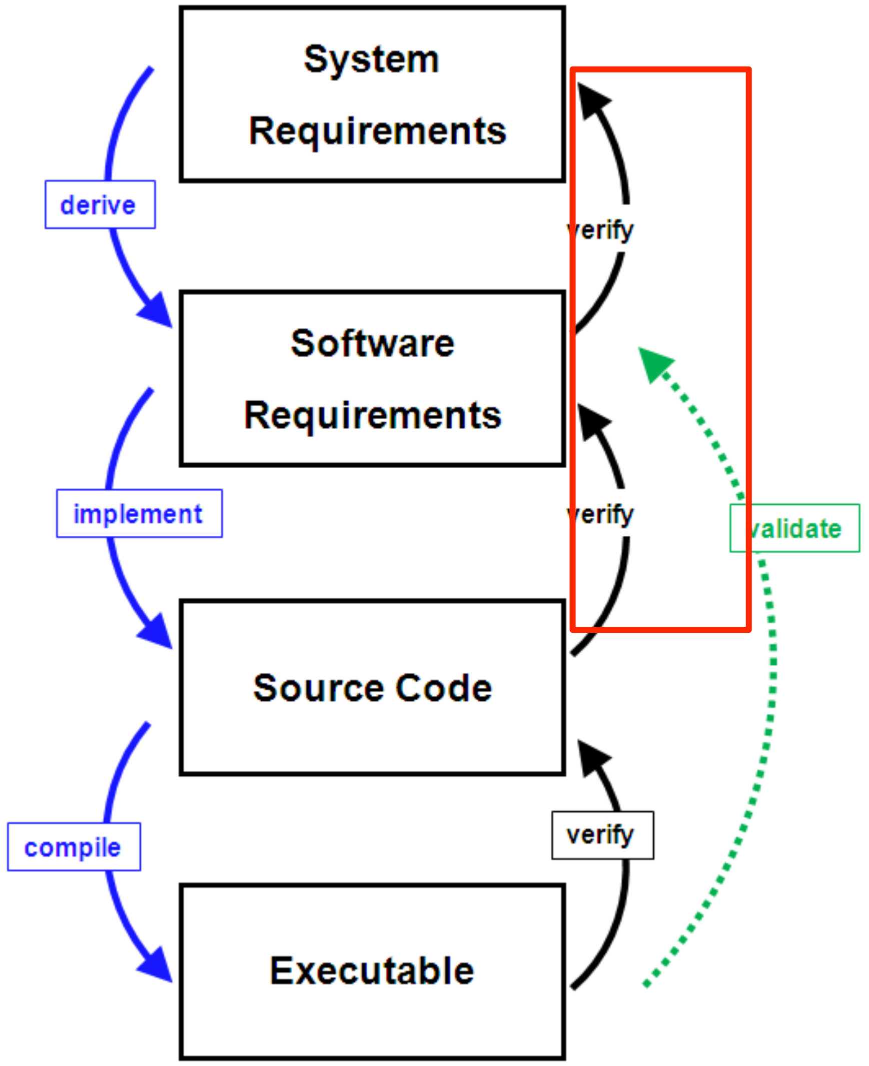
- This short presentation focuses on Section 6.0 of the FM supplement which is concerned with verification
- Important to understand that “verification” in DO 178B means evaluation of the outputs of a process step for consistency and correctness wrt to the inputs to this process step, e.g.,
  - Verification of the S/W requirements
  - Verification of the architecture
  - Verification of the source code
  - Verification of the executable code

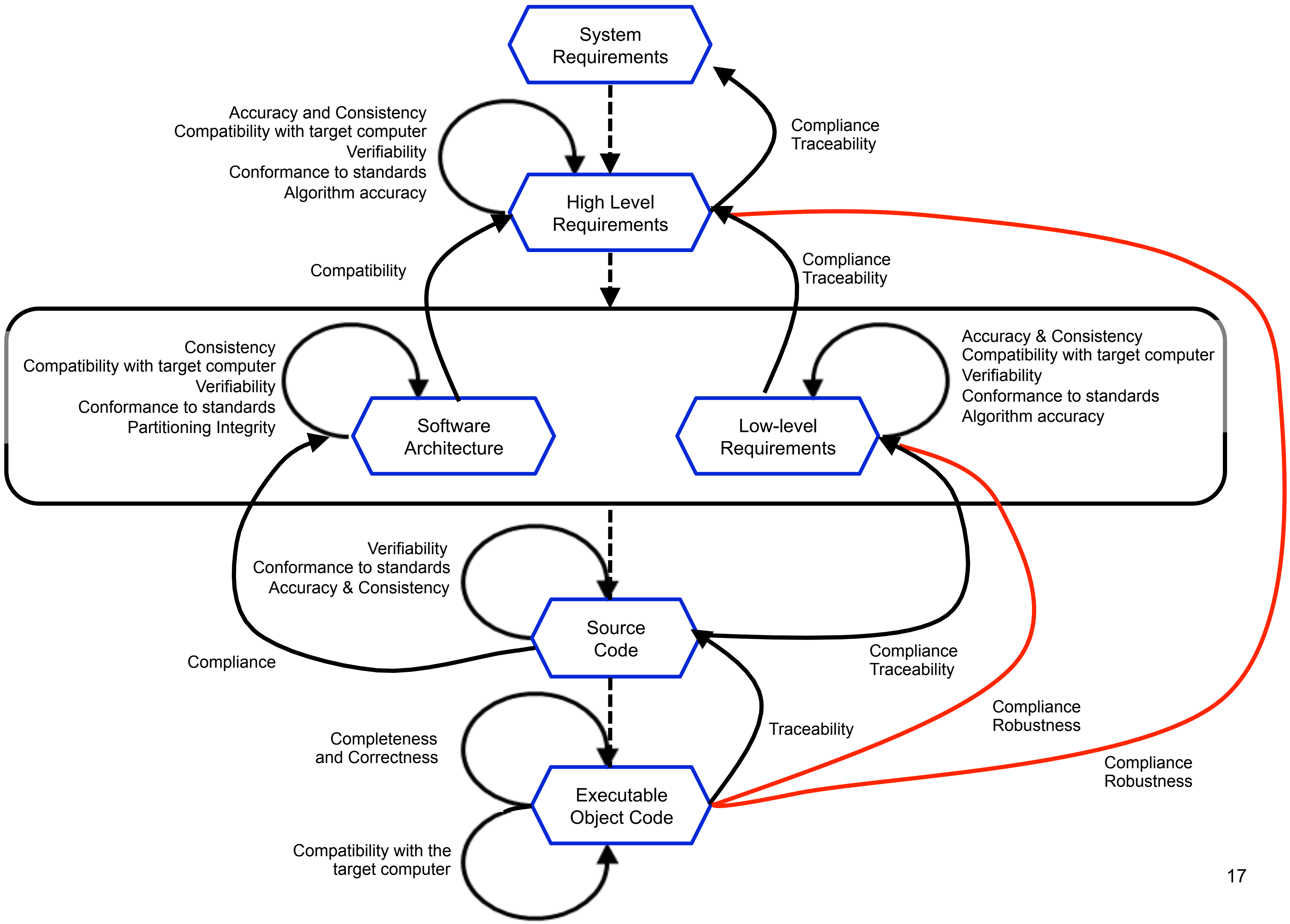
conventionally a matter of review and analysis

conventionally a matter of test

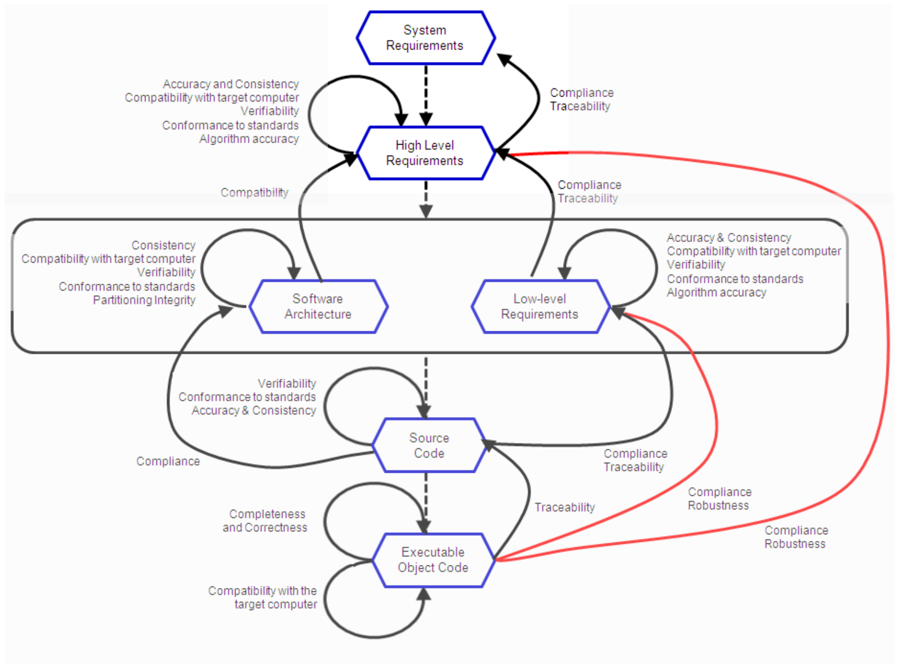
Verification vs. Validation

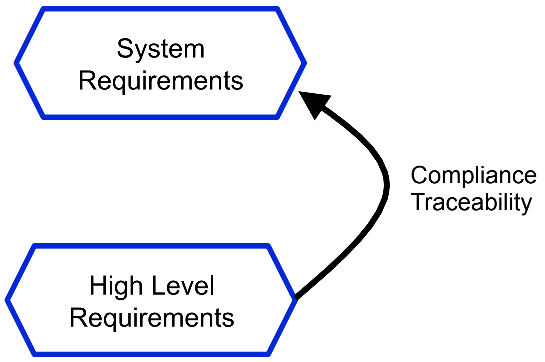
- These two terms have a very specific meaning in the context of DO 178B and several other standards (e.g., ISO 26262, EN 50128)




System Requirements
High Level Requirements
Accuracy and Consistency
Compatibility with target computer
Verifiability
Conformance to standards
Algorithm accuracy
Compliance
Traceability
Compatibility
Compliance
Traceability
Consistency
Compatibility with target computer
Verifiability
Conformance to standards
Partitioning Integrity
Software Architecture
Low-level Requirements
Accuracy & Consistency
Compatibility with target computer
Verifiability
Conformance to standards
Algorithm accuracy
Verifiability
Conformance to standards
Accuracy & Consistency
Source Code
Compliance
Compliance
Traceability
Traceability
Compliance
Robustness
Completeness
and Correctness
Executable Object Code
Compliance
Robustness
Compatibility with the
target computer

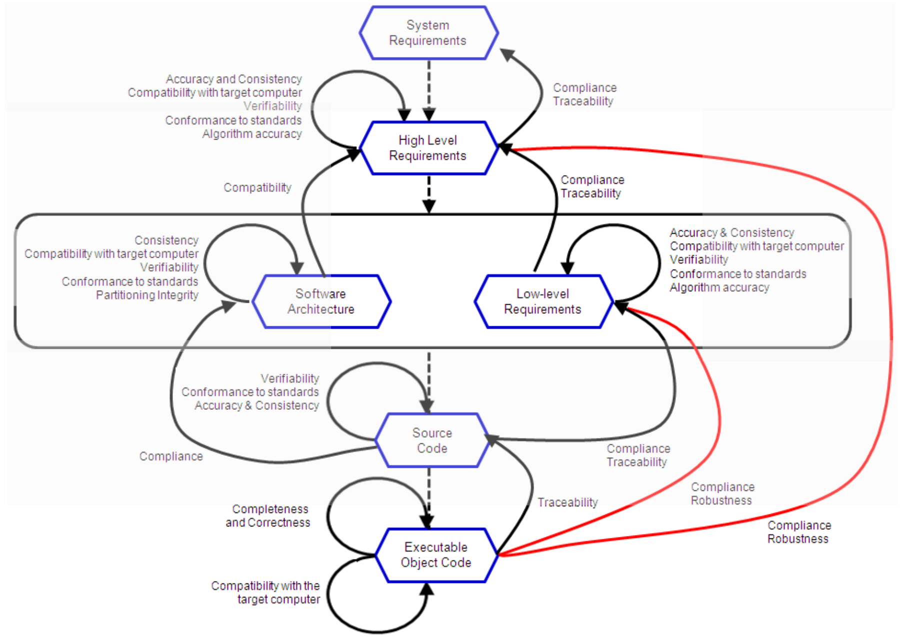
FM 6.0 – Sys Reqs ←→ HSRs

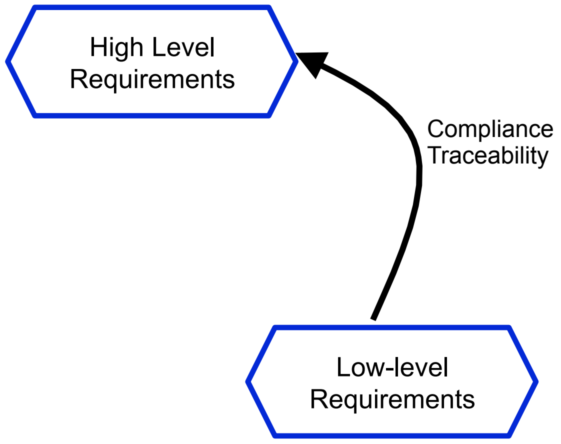




- For example, formal analysis may be used to show that the HSRs comply with the System Requirements and are traceable to the System Requirements

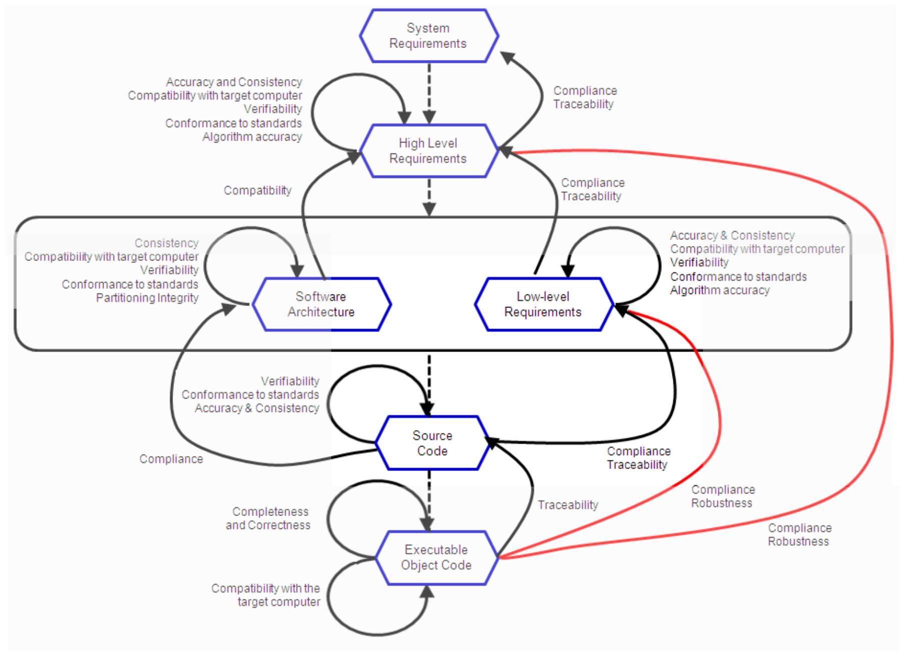
FM 6.0 – HSRs ←→ LSRs

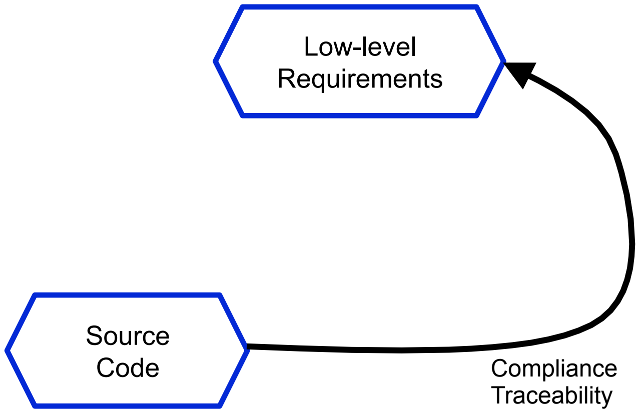




- Similarly, formal analysis may be used to show that the LSRs comply with the HSRs and are traceable to the HSRs

FM 6.0 – LSRs ←→ Source Code

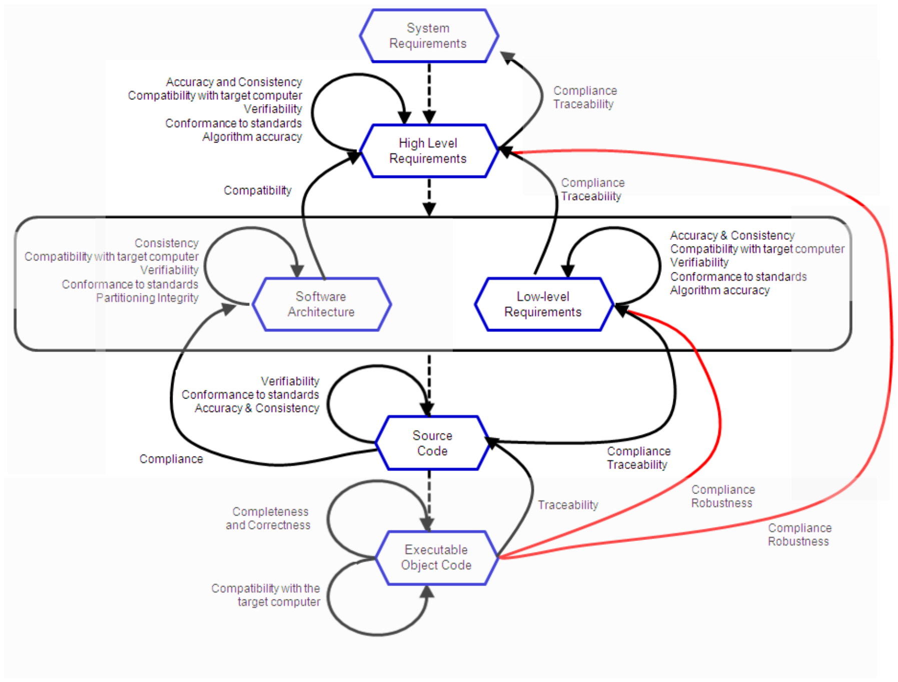


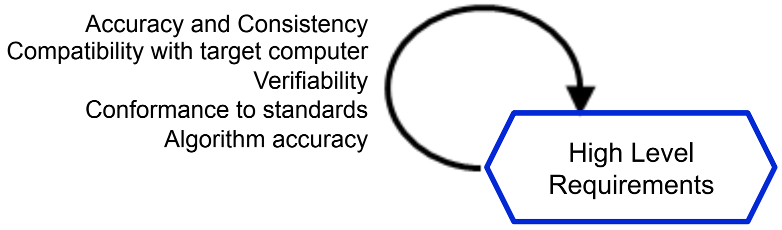


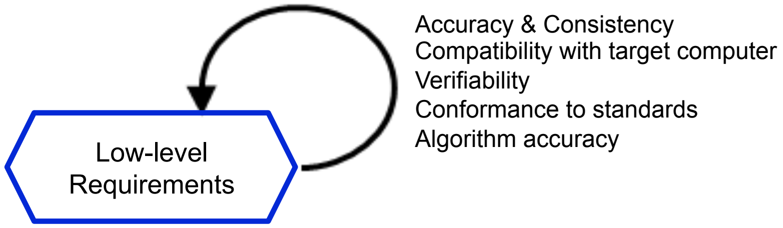
- Similarly, formal analysis may be used to show that the source code complies with the LSRs and is traceable to the LSRs

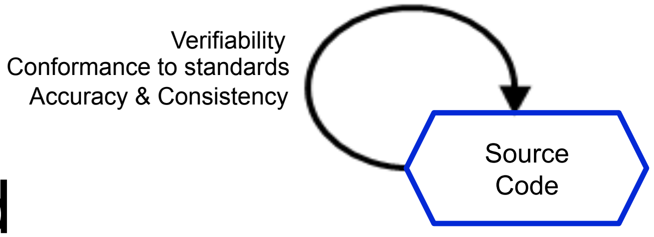
Critical Systems Labs
Strategic Insight for Safety

FM 6.0 – Consistency, etc.



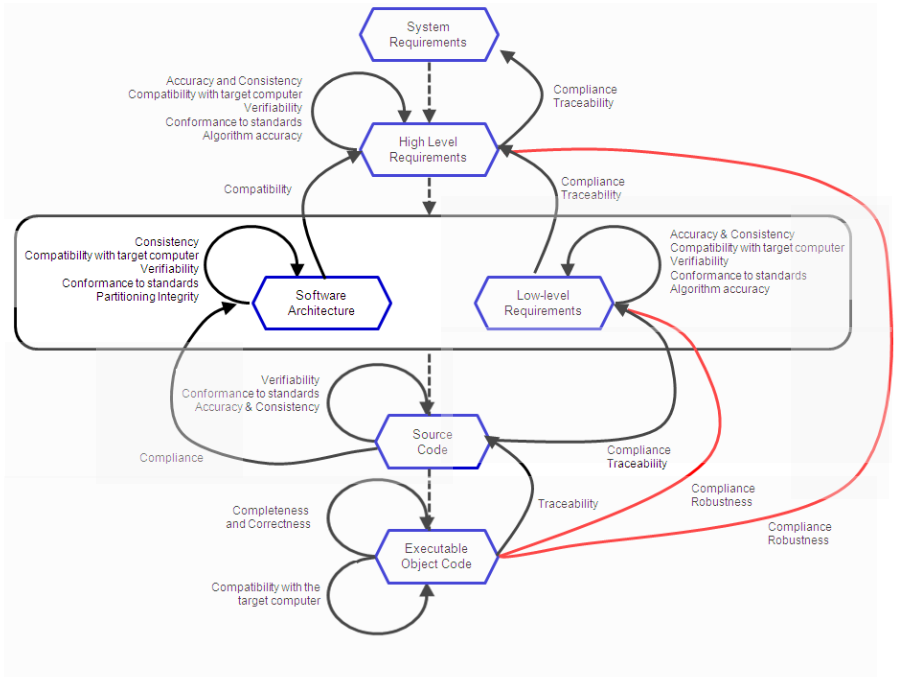


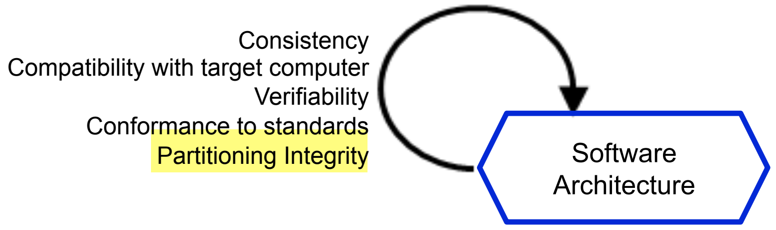




- In addition to compliance and traceability, formal analysis may be used to satisfy other verification objectives such as consistency

FM 6.0 – Partitioning



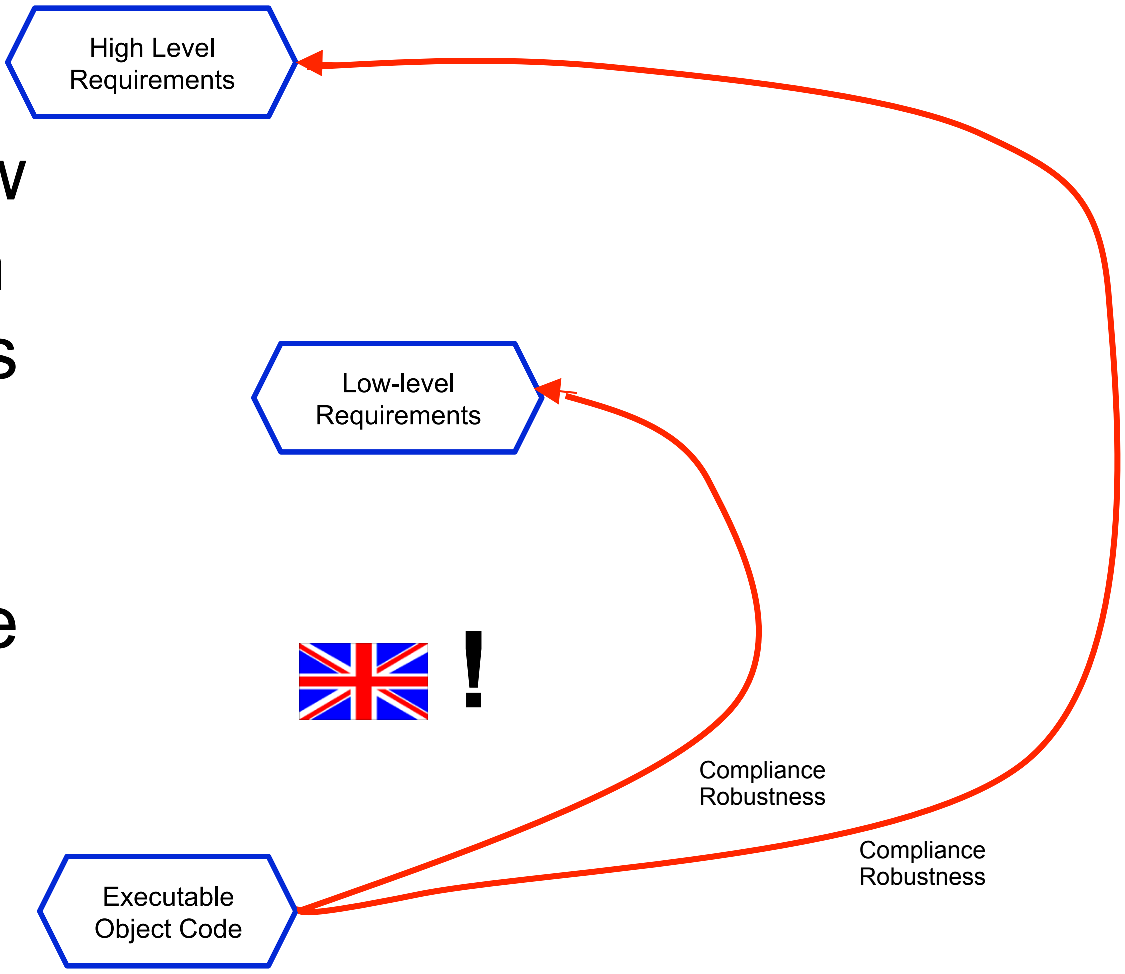


- Formal analysis may be used to demonstrate partitioning integrity

!

FM 6.5 - Formal Analysis of the Executable Object Code

- May be possible to show compliance with HSRs and LSRs involves formal analysis of the executable code using a formal model of the executable code



Critical Systems Labs
Strategic Insight for Safety

FM 6.0 - Software Verification Process

- There is a potential to at least partially satisfy most verification objectives using formal analysis

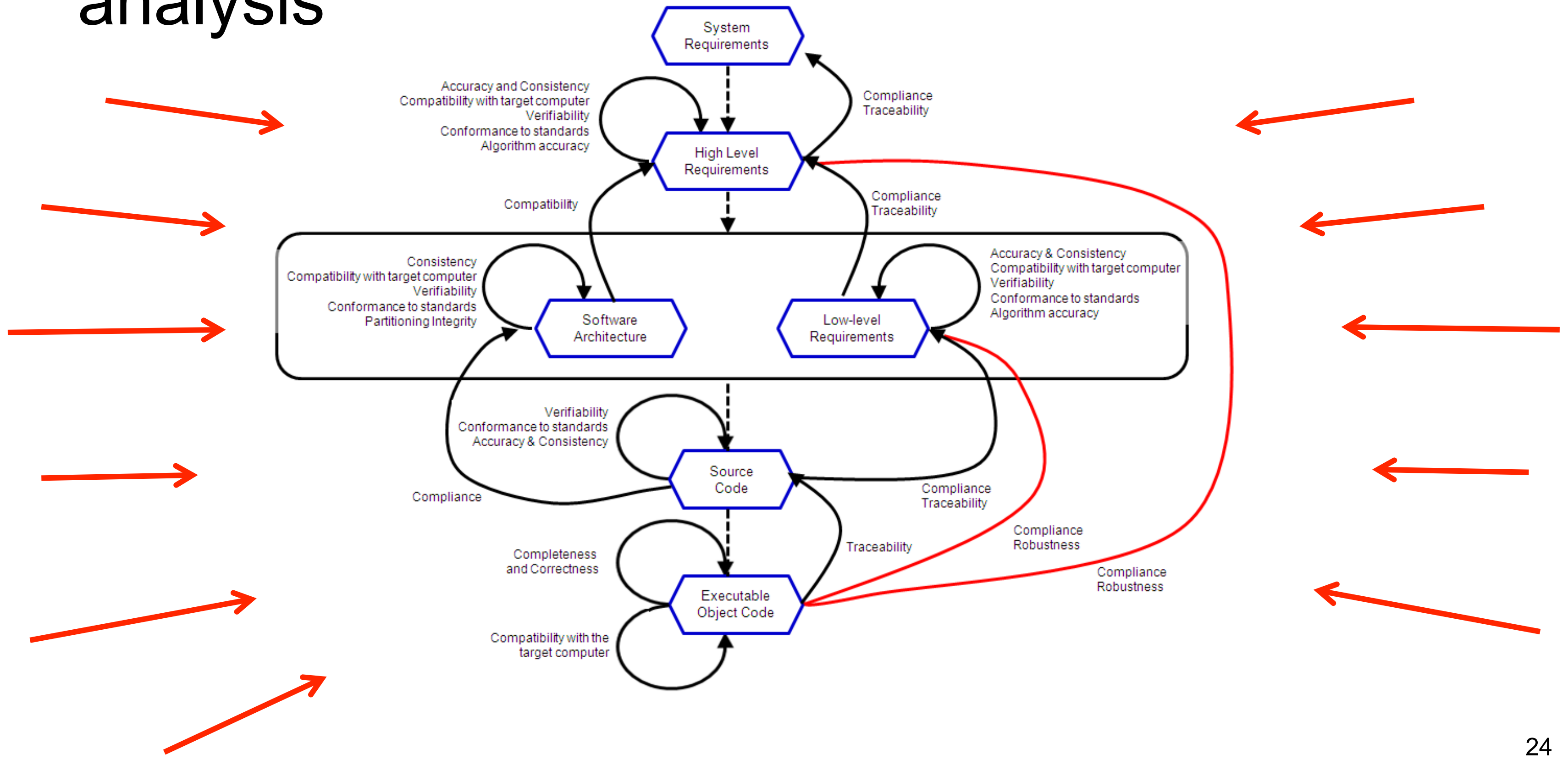


Table FM A-7

Verification Of Outputs of Verification Processes

FM 1	Formal analysis cases and procedures are correct.	FM.6.3.8a FM.6.3.8b	●	❍	❍		Software Verification Results	11.14	②	②	②	
FM 2	Formal analysis results are correct and discrepancies explained.	FM.6.3.8c	●	❍	❍		Software Verification Results	11.14	②	②	②	
FM 3	Coverage of high-level requirements is achieved.	FM.6.5.1.1	●	❍	❍	❍	Software Verification Results	11.14	②	②	②	②
FM 4	Coverage of low-level requirements is achieved.	FM.6.5.1.1	●	❍	❍		Software Verification Results	11.14	②	②	②	
FM 5-8	Complete coverage of each requirement is achieved. The set of requirements is complete. Unintended dataflow relationships are detected. Dead Code and Deactivated Code are detected.	FM.6.5.1.2 FM.6.5.1.3 FM.6.5.1.4 FM.6.5.1.5	●	●	❍		Software Verification Results	11.14	②	②	②	
FM 9	Formal method is correctly defined and sound.	FM 6.2	●	❍	❍	❍	Software Verification Results	11.14	②	②	②	②

Three Details of Special Interest

1. Soundness
2. Conservation Representation
3. Unintended Functionality (and Structural Coverage Analysis Resolution)

Excerpts from the FM Supplement

- FM 1.6.2: Any tool that supports the formal analysis should be assessed under the tool qualification guidance required by DO-178 and qualified where necessary.
- FM 6.2: The soundness of each *formal analysis* method should be justified. A sound method never asserts that a *property* is true when it may not be true.

Soundness

- … of the formalism
- … of the software tools
- … of axioms added to the pure formalism
- … of assumptions about the subject matter

"… and then at this point we replaced the decorated operational semantics with the angelic operational semantics …"

- Some worries …
  - While we may know what is/is not sound, what should be provided to a certification authority as objective evidence of soundness?
  - Moreover, some formal methods tools are not intended to be sound

Huh?

Three Details of Special Interest

1. Soundness
2. Conservation Representation
3. Unintended Functionality (and Structural Coverage Analysis Resolution)

FM 6.3 – Software Reviews and Analysis

- Requirement formalization correctness: If a requirement has been translated to a formal notation as the basis for using a formal analysis, then review or analysis should be used to demonstrate that the formal statement is a conservative representation of the informal requirement.

  Note: *If the gap between an informal statement of the requirement and its embodiment in a formal notation is too large then this may be difficult to review. The preciseness of formal notations is only an advantage when they maintain fidelity to the intent of the informal requirement.*

Critical Systems Labs
Strategic Insight for Safety

Conservative Representation

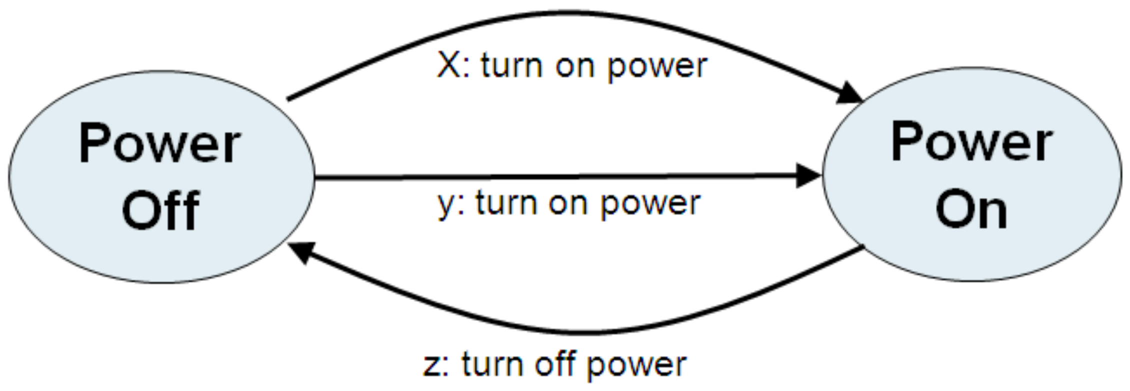
- Conservative representation - A simplified or abstract representation in a formal notation of life-cycle data (such as requirements or code) such that statements that are true for this representation will always be true for the life-cycle data itself. In many cases, analysis of a conservative representation will yield pessimistic results but never unsound ones. The representation chosen simplifies or makes possible the analysis.

Conservative Representation (+1)

- It is very common for a formal model to be less detailed than the life cycle artifact that it models
- However, missing details might invalid the results of formal analysis if the formal model is not a conservative representation
- If it is a conservative representation, then the everything derivable from the formal model by formal analysis is also true of the original artifact

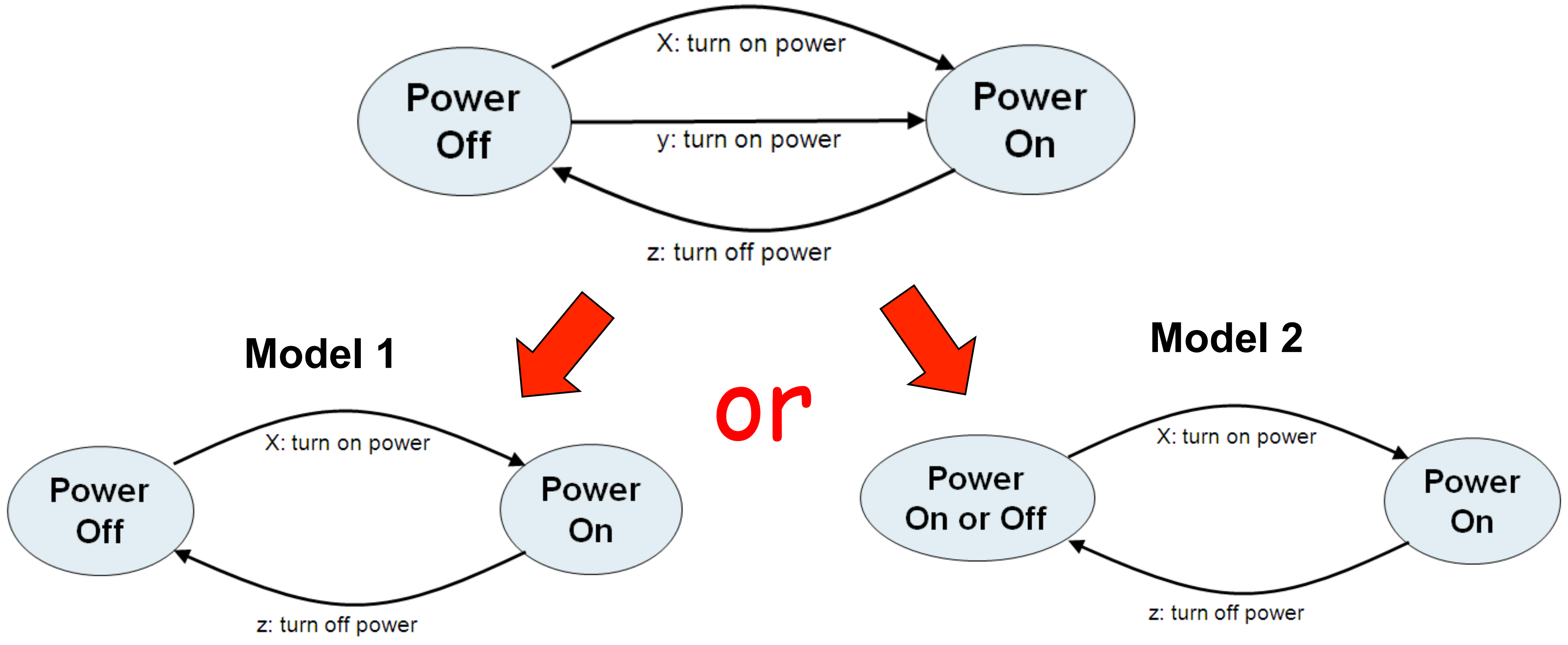
Conservative Representation (+2)

- Suppose that we have a system that controls a source of electrical power … at all times, either the power is on or the power is off
- Three different possible inputs to the system, x, y and z
- Responds to a sequential stream of inputs as follows …



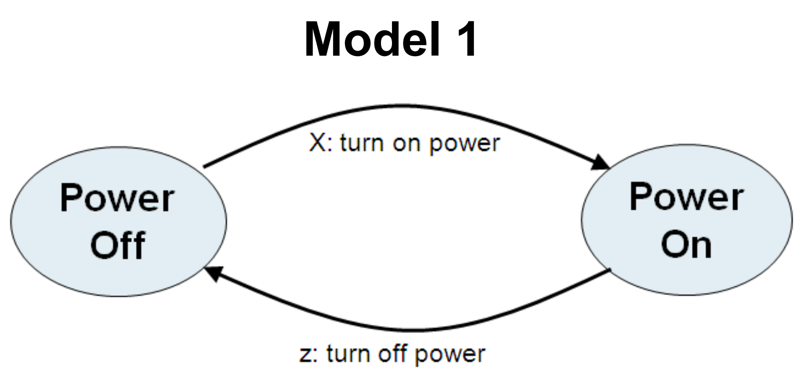
- Let' s assume that it is desirable to simplify the formal model by excluding any mention of input y

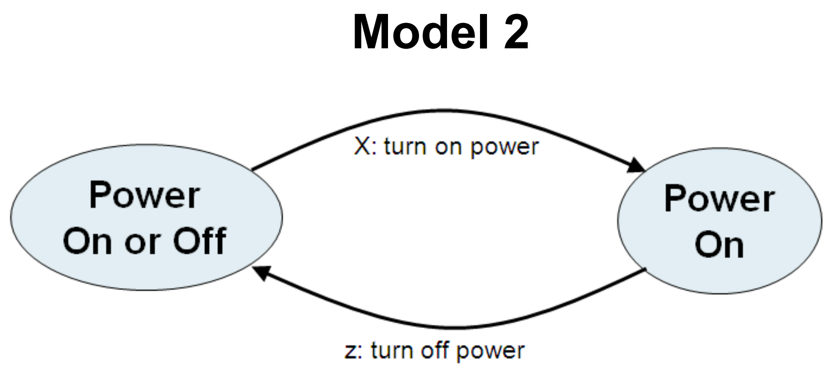
Conservative Representation (+3)



- Which of these two simplifications, Model 1 or Model 2, is a conservative representation of the original behaviour?

Conservative Representation (+4)





- Both formal models logically imply that the system will stay in the "power on" state after receiving input x until the next occurrence of input z … which is true ✓
- However, Model 1 also logically implies that the system will stay in the "power off" state after receiving input z until the next occurrence of input x … which is false! ✗

Critical Systems Labs
Strategic Insight for Safety

Two More Worries

- It' s not entirely clear what would be an acceptable way to show that a formal model is conservative representation of an informal artifact
- It is common to see formal models that are intentionally ***not*** conservative representations
  - … and the result is useful because it often finds problems

Critical Systems Labs
Strategic Insight for Safety

Three Details of Special Interest

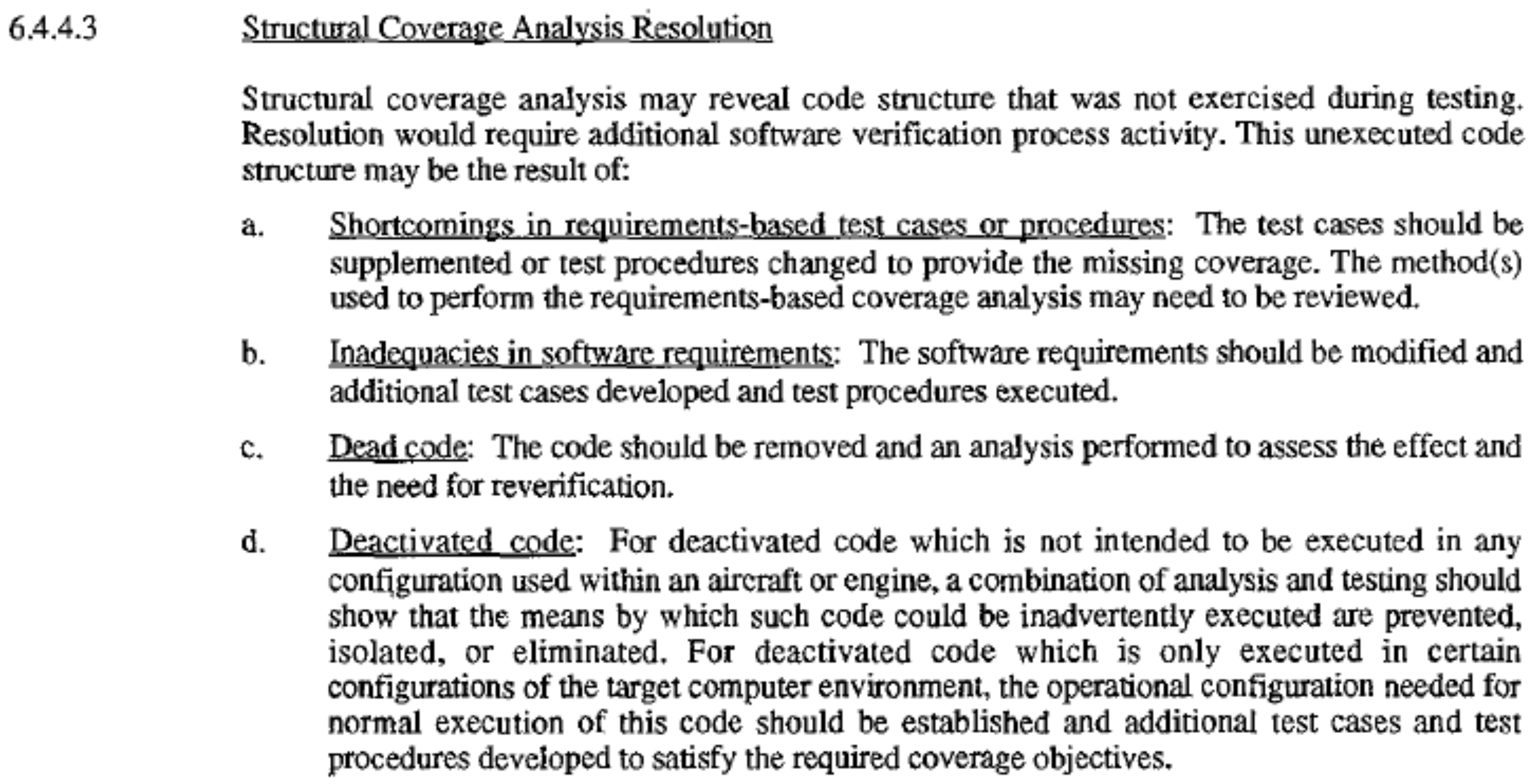
1. Soundness
2. Conservation Representation
3. Unintended Functionality (and Structural Coverage Analysis Resolution)

Critical Systems Labs
Strategic Insight for Safety

Unintended Functionality

- A very significant verification goal of DO 178B is to demonstrate the absence of unintended functionality in the S/W
  - a.k.a. undocumented code, unexpected code structure
- There are several provisions in DO 178B that address unintended functionality, including "Structural Coverage Analysis Resolution" in combination with coverage metrics such as MC/DC

Structural Coverage Analysis Resolution



6.4.4.3 Structural Coverage Analysis Resolution

Structural coverage analysis may reveal code structure that was not exercised during testing. Resolution would require additional software verification process activity. This unexecuted code structure may be the result of:

a. Shortcomings in requirements-based test cases or procedures: The test cases should be supplemented or test procedures changed to provide the missing coverage. The method(s) used to perform the requirements-based coverage analysis may need to be reviewed.

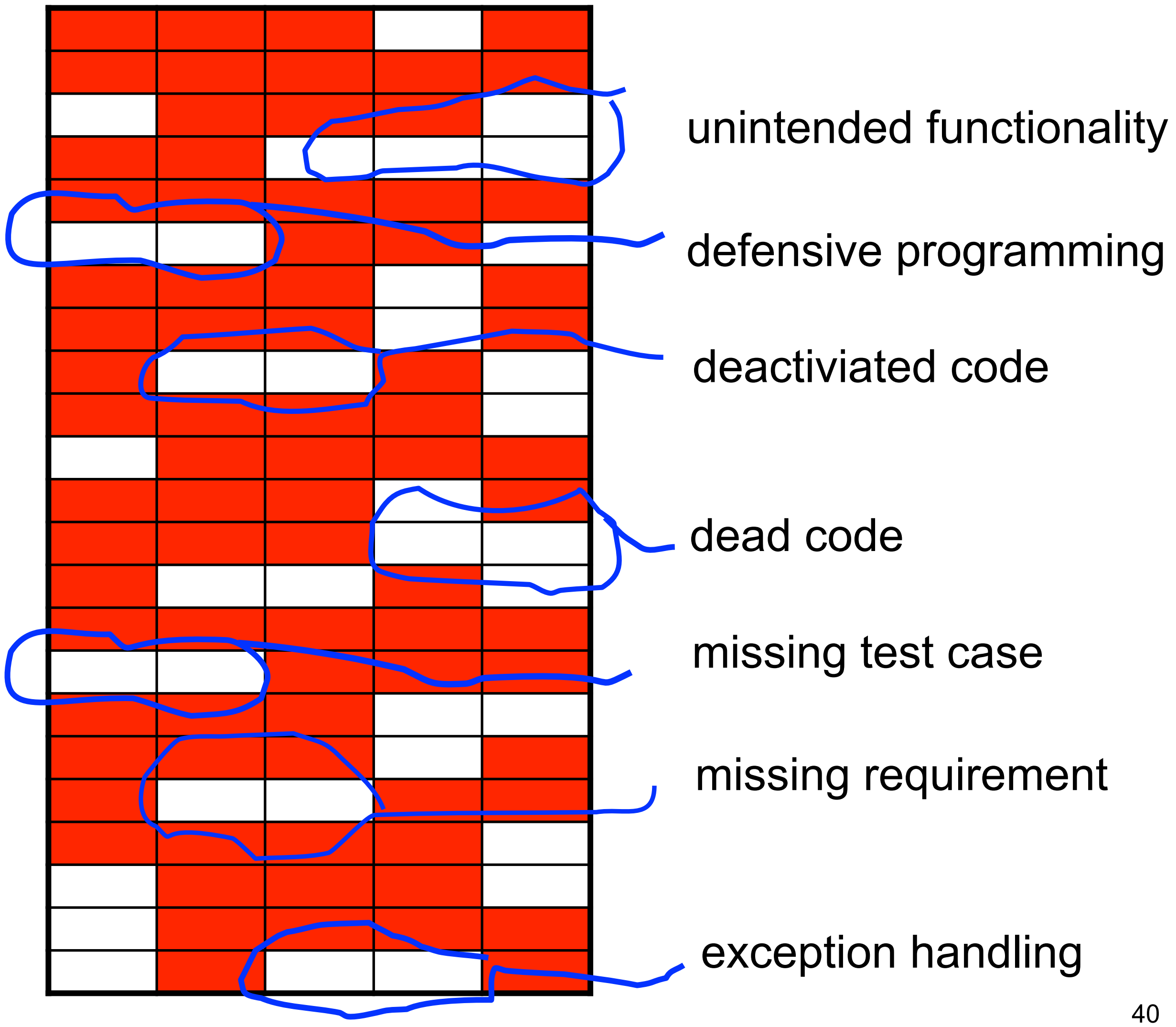
b. Inadequacies in software requirements: The software requirements should be modified and additional test cases developed and test procedures executed.

c. Dead code: The code should be removed and an analysis performed to assess the effect and the need for reverification.

d. Deactivated code: For deactivated code which is not intended to be executed in any configuration used within an aircraft or engine, a combination of analysis and testing should show that the means by which such code could be inadvertently executed are prevented, isolated, or eliminated. For deactivated code which is only executed in certain configurations of the target computer environment, the operational configuration needed for normal execution of this code should be established and additional test cases and test procedures developed to satisfy the required coverage objectives.

Structural Coverage Analysis

- There are a variety of reasons for “gaps” in the coverage obtained using a specific coverage criteria



Another Worry

- Detecting unintended functionality does not appear to be a common use of formal methods, and it is not clear (to some of us) how this could be done in a practical way using current tools and methods
- If there is nothing to replace "structural coverage analysis resolution", then you might be obliged to do almost as much testing as you would have done without any use of formal methods

FM 12.3.6 - Coverage Analysis …

- This section expresses the conclusion of SG6 that, at the time of writing the FM supplement, "it was only practical to establish coverage for a software component using either a *formal method* approach or a test based approach but not a combination of the two"
- However, it also recognizes that there is a possibility that there might someday be a practical approach where coverage can be determined even when using a combination of formal analysis and testing
- Hopefully, formal methods researchers will take up the challenge providing a means of achieving coverage with a mixed approach

Worries, worries and more worries

- Are such “worries” are not a shortcoming of the FM supplement?
- No, they are a consequence of a fundamental desire to be “technology neutral”, i.e.,
  - …. accommodating all established approaches to formal methods and hopefully many future approaches

Summary

- The DO 178C Formal Methods supplement is a great opportunity for bringing formal methods into the “mainstream” of verification methodology for airborne software
- Very likely to serve as a model for how formal methods should be used as part of a certification effort for other domains
- However, there are a number of questions that need to be addressed about how this guidance would be applied for specific approaches to FM

Critical Systems Labs
Strategic Insight for Safety

Some questions (for TP' ers who aspire to verify airborne S/W for certification)

- What should be required to accurately identify a particular approach to theorem-proving?
- What should be required to demonstrate that a particular theorem-proving approach is sound?
- Is the notion of "conservative representation" sufficient for a theorem-proving approach to ensure that the formalization of a verification problem does not admit false results?
- Should there be limitations on the number and complexity of requirements addressed by a single verification result obtained by means of theorem-proving?
- How should verification results obtained by means of theorem-proving be documented?
- How can theorem-proving demonstrate the absence of unintended functionality?

Workshop on Theorem Proving in Certification 6-7 December 2010, Cambridge, UK

http://www.cl.cam.ac.uk/~mjcg/FMStandardsWorkshop.html

- Follow-up workshop in Fall 2011 ?