# Software Certification and DO-178C

## SC-205 update

**AFRL Safe & Secure Systems & Software Symposium (S5)**
**June 3, 2009**

**Darren Cofer**

Rockwell Collins

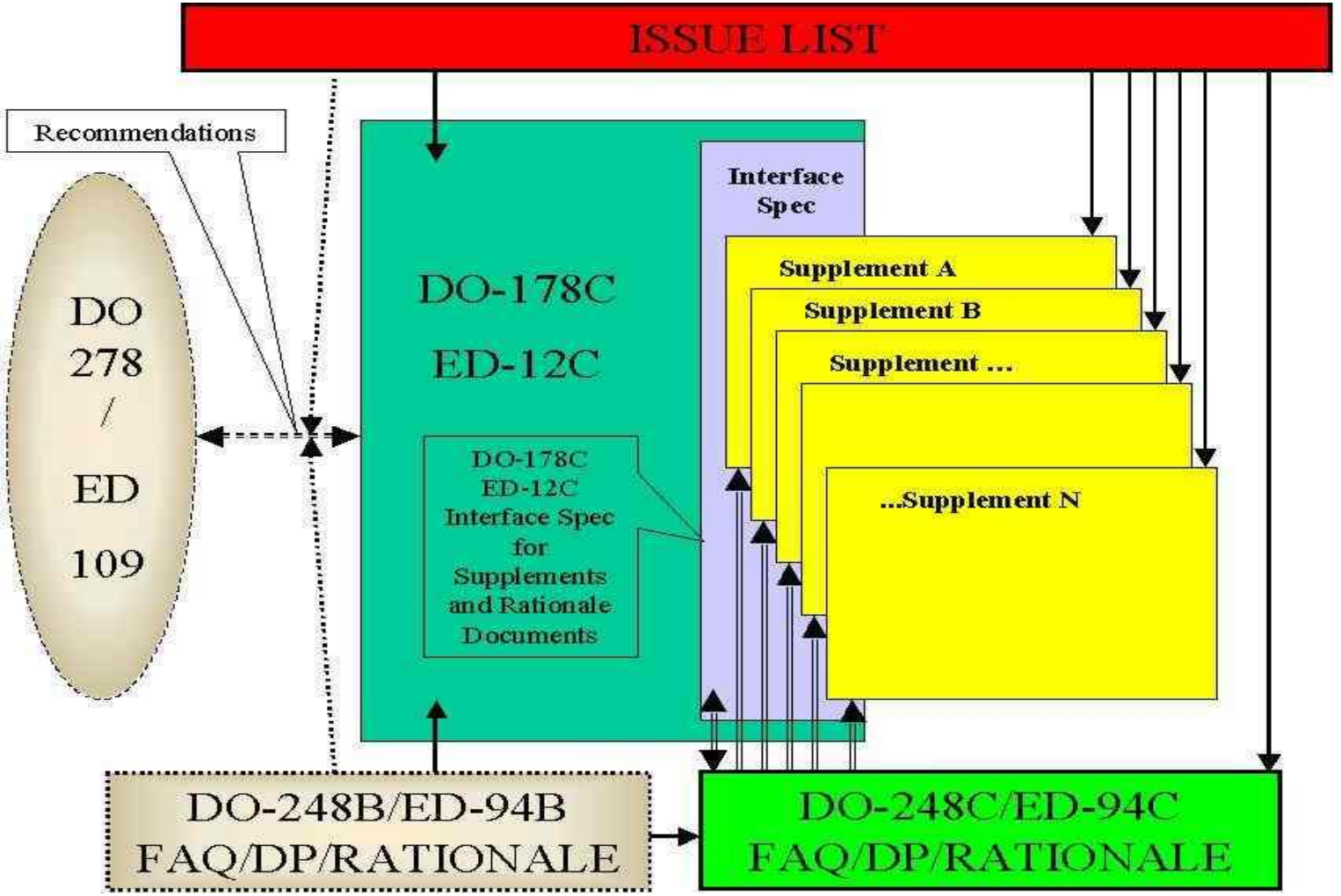
# SC-205 background

- Committee chartered by RTCA to update DO-178B
- Joint with Europe
  - RTCA : SC-205 : DO-178B : FAA ::
    EUROCAE : WG-71 : ED-12B : EASA
- How does it work?
  - Meetings (10 plenary so far)
  - Web site
  - Information Papers (IP)
  - Discussion, negotiation, arguing, voting
- Players
  - Boeing, Airbus, Saab, Embraer…
  - Rolls Royce, P&W…
  - Rockwell Collins, Honeywell, Thales, Goodrich, GE…
  - Mathworks, Esterel…
  - FAA, EASA, NASA…
  - Etc. (freelance DERs, smaller suppliers)

# DO-178C objectives

- What is it?
  - "Software Considerations in Airborne Systems"
  - developed by stakeholders (airframers, OEMs, cert authorities)
  - consensus recommendations ("I can live with it.")
  - certification guidance related to software: if accepted, government certification authorities agree that an applicant can use it to obtain certification (of airplane/engine)
- Terms of Reference
  - marching orders from RTCA & EUROCAE (2005)
  - minimize changes to core document, yet…
  - update to accommodate 15+ years of SW experience
- Technologies to address in "supplements"
  - OO, FM, MBD
- Air/ground synergy?
  - DO-278
- Also: rationale, consolidation, issues, errata
  - DO-248

# DO-178C structure



# DO-178C: Expected changes

- Objectives: New guidance for emerging SW trends and technologies, document rationale, address gaps/inconsistencies
- Schedule: Committee work is scheduled for completion 2010/Q1
- Structure: Minimize changes to core document
  - Maintain current objective-based, technology-neutral approach
  - *Supplements* to address specific technologies
  - Coordination with DO-278, but no merger of air/ground guidance
- Tool Qualification supplement: Likely to increase tool qual. effort
  - More detailed, stand-alone document, complete with objectives
  - 3 tool criteria (1 = development, 3 = verification, 2 = something in between)
  - [criteria] + [SW level] => tool qual. level (1-5) => required objectives
- Model-Based Design & Verification supplement: Model = Requirements
  - New guidance for model execution/simulation
  - Attempting to define model coverage metrics
- Object-Oriented Design supplement: Nothing significant yet
- Formal Methods supplement: Facilitate use of FM for SW verification
  - Facilitate applicant/certifier communication (definitions, expected evidence)
  - Define new objectives/activities/documentation (abstractions, assumptions)
  - Avoid common errors (false hypotheses)

# Tool Qualification

- Replace 1/2 page of text in DO-178B with 50+ page supplement
- Introduce 5 levels of tool qualification level
  - roughly correspond to development tools for SW level A-D + verification tool
- Introduce 3 criteria for determining TQL
  1. SW development tool (could insert error)
  2. A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of verification process(es) other than that automated by the tool, or development process(es) which could have an impact on the airborne software.
  3. SW verification tool (could fail to detect error)
- Why do we care? If criteria 2 is applied to FM tools it could make it difficult to use open/academic analysis engines

# Objectives of Formal Methods subgroup (SG-6)

- Identify scope of FM
  - What activities can be satisfied
  - Focus is on verification (DO-178 section 6)
  - Ensure that requirements for (some) testing is preserved
- Facilitate communication between applicants and certification authorities
  - What evidence should be expected for satisfying objectives
  - What new process documentation is needed
  - What additional/different activities are needed
- Identify areas deserving of particular scrutiny when FM are used
  - Help to avoid common errors
  - What questions should be asked
- Facilitate use of FM in aerospace community
  - Don't impose higher burdens than traditional processes
  - But still keep the "bar" high enough

# Section 1 – INTRODUCTION

- All new text
  - explanatory information
  - not *guidance*
- Characteristics of FM
  - models + analysis
  - use may be limited to particular requirements, systems, activities
- Formal modeling
  - unambiguous, mathematically defined syntax and semantics
- Formal analysis
  - guarantee *properties* of software systems
  - emphasis on *soundness* of method
- FM as verification technique
  - Provide guidance for using formal methods to meet verification objectives of DO-178, either fully, or in conjunction with additional verification activities, including suitable complementary testing on the target hardware

Rockwell Collins

## Section 2 – SYSTEM ASPECTS RELATING TO SW DEV.
## Section 3 – SOFTWARE LIFE CYCLE

- unchanged by FM Supplement

# Section 4 – SOFTWARE PLANNING PROCESS

- PSAC will identify how FM will be used and what evidence will be provided
- SDP will identify development steps using FM
- SVP will identify:
  - analysis methods
  - assumptions and impact
  - how verification techniques will be combined to meet objectives

# Section 5 – SOFTWARE DEVELOPMENT PROCESS

- Formalization of requirements and design artifacts
  - Not all requirements need to be formally modeled
  - Requires sufficient definition to verify properties of artifacts being analyzed
  - Errors may be found as a byproduct of the formalization process

# Section 6 – SOFTWARE VERIFICATION PROCESS

- Focus of FM guidance is on Verification Process
- General guidance:
  - All *formal notations* used should be verified to ensure that they have unambiguous, mathematically defined syntax and semantics.
  - The soundness of each *formal analysis* method should be verified. A sound method never asserts that a *property* is true when it may not be true.
  - All assumptions related to the *formal analysis* should be described and justified (e.g. those associated with the target computer, or those about the data range limits).

# Section 6 – SOFTWARE VERIFICATION PROCESS

**Compliance**: If the life cycle data items that comprise the inputs and outputs of a software development process are formally modeled, then *formal analysis* can be used to verify compliance. Compliance can be demonstrated by showing that the output satisfies the input. *Formal methods* cannot show that a derived requirement is correctly defined and has a reason for existing; this must be achieved by review.

**Accuracy**: *Formal notations* are by definition precise and unambiguous, and can be used to demonstrate accuracy of the representation of a life cycle data item.

**Consistency**: Life cycle data items that are formally expressed can be checked for consistency (the absence of conflicts). If a set of formal statements is found to be logically inconsistent then the inconsistencies must be addressed before any subsequent analysis is conducted.

**Compatibility with the target computer**: *Formal analysis* can be used to detect potential conflicts between a formal description of the target computer and the life cycle data item.

**Verifiability**: Being able to express a requirement in the *formal notation* defined in the software verification plan is evidence of verifiability in the same way as being able to define a test case.

Note: *Requirements involving "always/never" cannot in general be verified by a finite set of test cases, but may be verified by formal analysis.*

**Conformance to standards**: Life cycle data items that are formally expressed must be compliant with any standards defining the formalism.

Note: *Invalid results will be obtained if ill-formed requirements are allowed. Since formal notations have by definition a well-defined syntax, automated syntax checkers are appropriate for verifying that the formally stated requirements are well-formed with respect to syntax.*

**Traceability**: Traceability ensures that all input requirements have been developed into lower level requirements or source code. Traceability from the inputs of a process to its outputs can be demonstrated by verifying with a *formal analysis* that the outputs of the process fully satisfy its inputs. Traceability from the outputs of a process to its inputs can be demonstrated by verifying with a *formal analysis* that each output data item is necessary to satisfy some input data item.

Note: *Where output data items have been logically derived from input data items by refinement, traceability becomes implicit.*

**Algorithm aspects**: If life cycle data items are formally modeled, then algorithmic aspects can be checked using *formal 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.*

new objective

# Section 6 – SOFTWARE VERIFICATION PROCESS

- Formal Analysis of Executable Object Code
  - In 178B this is accomplished by testing, and is one of the main objects of the verification process
  - Requirements-based test + structural coverage measurements
  - 178C allows some of this to be replaced by analysis, but some testing must still be performed
  - Based on Airbus A380 experience
- Principles
  - Complete coverage of each requirement
  - Complete coverage of the set of requirements
  - Detection of unexpected dataflow relationships (dependencies)
  - Detection of dead/deactivated code

## Section 7 – SW CONFIG. MANAGEMENT PROCESS
## Section 8 – SW QUALITY ASSURANCE PROCESS
## Section 9 – CERTIFICATION LIAISON PROCESS
## Section 10 – OVERVIEW OF A/C & ENGINE CERT.

- unchanged by FM Supplement

# Section 11 – SOFTWARE LIFE CYCLE DATA

- SW requirements, design, and coding standards
  - Specify formal notations to be used
- Verification procedures
  - Define the properties to be verified
  - Describe the analysis procedure to be used
  - Define the environment of the formal analysis (assumptions, constraints)

# Section 12 – ADDITIONAL CONSIDERATIONS

- Coverage analysis when using a *combination* of formal methods and testing in a single component
  - We don't know how to do this now
- This is really just guidelines for future work. Would need to address:
  - Requirements coverage
  - Structural coverage of code

# ANNEX A – PROCESS OBJECTIVE TABLES

- Table A-3,4,5 new objectives

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **FM 8** | Formal analysis cases and procedures are correct. | FM.6.2<br>FM.6.3.7a<br>FM.6.3.7b | ● | ❍ | ❍ | | Software Verification Results | 11.14 | ② | ② | ② | |
| **FM 9** | Formal analysis results are correct and discrepancies explained. | FM.6.3.7c | ● | ❍ | ❍ | | Software Verification Results | 11.14 | ② | ② | ② | |
| **FM 10** | Requirements formalization is correct. | FM 6.3i | ● | ● | ❍ | ❍ | Software Verification Results | 11.14 | ② | ② | ② | ② |

# ANNEX A – PROCESS OBJECTIVE TABLES

- Table A-7 new objectives
  - replacing objectives 1-8 if object code analysis instead of test

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **FM 1** | Formal analysis cases and procedures are correct. | FM.6.2<br>FM.6.3.8a<br>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<br>Completeness of the Set of Requirements<br>Detection of Unintended Dataflow Relationships<br>Detection of Dead Code and Deactivated Code | FM.6.5.1.2<br>FM.6.5.1.3<br>FM.6.5.1.4<br>FM.6.5.1.5 | ● | ● | ❍ | | Software Verification Results | 11.14 | ② | ② | ② | |

# ANNEX B – GLOSSARY OF TERMS

- Terms to be defined…
  - Conservative representation
  - Formal Analysis – replacing general definition of 'analysis'
  - Formal analysis cases – replacing test cases
  - Formal Methods
  - Formal Model
  - Formal Notation – language, unambiguous mathematically defined syntax/semantics
  - Information Flow
  - Property – formalized requirement
  - Soundness – applied to underlying methods, not tools
  - Well-formed

Rockwell Collins

# Next actions

- Next meeting:  Hartford CT in June
  - Two more meetings scheduled through Q1 2010
- Next documents:  Vote on complete FM supplement (6-12)
- Original completion date:  12/2008
- Revised completion date:  3/2010 ???