Model-Based Development & Verification Supplement to DO-178C

Are we there yet?

Mark Lillis
mark.lillis@goodrich.com

National Software/AEH Conference, September 2011
Overview

► What is the MBDV Supplement
  ▪ Background
  ▪ Introduction

► How does it link to Core DO-178C
  ▪ General Supplement format
  ▪ Planning & Annex Tables

► How does it link to other supplements
  ▪ More on Planning

► When is it mandatory
  ▪ Another review of Planning

► What are the major points
  ▪ Major Section Highlights
  ▪ Final Thoughts
Disclaimer

► The Model-Based supplement is not yet approved by the RTCA/EUROCAE committee.

► At the time of this presentation generation, some text has not been fully approved by the plenary committee vote but is fully approved by the sub-group and fully approved by a non-binding virtual plenary vote (blue text).

► Some text quoted in this presentation copy found on the CD may have been approved or revised. Further, until this document goes thru the public FRAC comment process & resolution stage, everything is still a draft.

► While close coordination with CAST has been maintained throughout this process, the supplement does not constitute regulatory policy until it is recognized in an Order, CRI, AC, etc...
What is the MBDV Supplement

► Brief Background on the MBDV Supplement
  ▪ What’s its purpose?
  ▪ Who is working on it?

► Introduction
  ▪ What’s covered by the Supplement?
  ▪ Why is it needed?
  ▪ Document Layout
Background

Purpose

- The Model Based Development & Verification Supplement is a companion document to the soon to be released DO-178C
- The MBDV Supplement is the joint work of Sub-Group 4 of RTCA SC-205 and EUROCAE WG-63
- The Supplement covers specific additional considerations that must be addressed when the design involves the use of Models
Background

► Working Team

- **Mark Lillis**, Pump & Engine Control Systems, Goodrich, West Hartford, CT
  - Co-Chair of SG-4

  - Co-Chair of SG-4

- **Martha C. Blankenberger**, Product Airworthiness, Rolls-Royce Corp., Indianapolis, IN
  - SG-4 Secretary

- **Patty (Bartels) Bath**, Esterline AVISTA, Platteville, WI
  - SG-4 Editorial Committee Representative
Working Team

► 27 meetings lasting 3-5 days each
► 90+ Attendees
► 25 Core Members representing a cross-section of industry

Goodrich, EADS-APSYS, Rolls-Royce, Esterline AVISTA, Cessna, Mathworks, Esterel Technologies, Rockwell Collins France

Hamilton Sundstrand, Rockwell Collins, Aero Engine Controls, Aeroconseil, EADS-MAS, Honeywell, NewTec GMBH, SAGEM DS

Embraer, EASA, ANAC Brazil, Atos Origin, Woodward, Airbus, Boeing
Introduction

What’s Covered by the Supplement

- Model-based development and verification technology involves methods and techniques to represent requirements in the form of a model
- These models are typically a graphical model
- They facilitate the development and/or verification of software
Two Words

► The entire MBDV Supplement can be summarized in just two words
  ▪ Requirements
  ▪ Simulation

► The two most important words are
  ▪ Planning
  ▪ Standards

► For many, it will be about just one word
  ▪ Verification
Glossary: Model

Not all pictures and figures meet the definition of a Model

Model - An abstract representation of a given set of aspects of a system that is used for analysis, verification, simulation, code generation, or any combination thereof. A model should be unambiguous, regardless of its level of abstraction.

Note 1: If the representation is a diagram that is ambiguous in its interpretation, this is not considered to be a model.

Note 2: The “given set of aspects of a system” may contain all aspects of the system or only a subset.
Introduction

Key principles:

- Models can represent high level and/or low level requirements
- More than one type of model may be used within a development or verification process
- Models are derived from, traced to, and verified against higher level requirements
- Simulation may utilize models to meet development and verification objectives
- Test is always required to ensure compatibility of the software with target hardware and to fully verify the understanding of the relationship between source and object code
Introduction

► Models are just part of the solution...
  ▪ The model-based techniques are rarely, if ever, used as the sole means to develop or verify software
  ▪ Model-based techniques may not be the optimum choice for all requirements
Why is the supplement needed

► The use of model-based development and verification technology in safety-critical airborne applications pre-dates DO-178B

► No special guidance was included in DO-178B to address the use of models

► Tools and techniques have evolved

► Auto-code generation, simulation, and test automation, led to inconsistent mapping of DO-178B objectives

► Consequently, there was a need to provide guidance for applicants and certification authorities to facilitate the use of models in the development and verification processes
Document Layout

- Document Layout

The MBDV supplement is laid-out to mirror the core DO-178C

- Document sections are prefixed with “MB”
  - Example: MB.5.1.1, Table MB.A-7
  - New sections are added at the end of existing 178C sections i.e. MB.6.8 *Model Simulation*

- MBDV supplement also contains Annex tables (Annex MB.C.x) to link with DO-278A Annex A tables

- Presently, the draft supplement contains FAQ’s and Discussion Papers (DP’s) in Appendix MB.B
  - These may move before final publication
## 6 SOFTWARE VERIFICATION PROCESS

### 6.1 Purpose of Software Verification

### 6.2 Overview of Software Verification Process Activities

### 6.3 Software Verification Processes

- **MB.6.0** SOFTWARE VERIFICATION PROCESSES
  - MB.6.1 Purpose of Software Verification
  - MB.6.2 Overview of Software Verification Process Activities
  - MB.6.3 Software Reviews and Analyses
    - MB.6.3.1 Reviews and Analyses of the High-Level Requirements
    - MB.6.3.2 Reviews and Analyses of the Low-Level Requirements
    - MB.6.3.3 Reviews and Analyses of the Software Architecture
    - MB.6.3.4 Reviews and Analyses of the Source Code
    - MB.6.3.5 Reviews and Analyses of the Outputs of the Integration Process
  - MB.6.4 Software Testing
  - MB.6.5 Software Verification Process Traceability
  - MB.6.6 Verification of Parameter Data Items
  - MB.6.7 Model Coverage Analysis for Design Models
    - MB.6.7.1 Model Coverage Analysis Criteria
    - MB.6.7.2 Model Coverage Analysis Resolution
  - MB.6.8 Model Simulation
    - MB.6.8.1 Model Simulation for Verification of the Model
    - MB.6.8.2 Model Simulation for Verification of the Executable Object Code
    - MB.6.8.3 Simulation Cases, Procedures and Results
      - MB.6.8.3.1 Development of Simulation Cases, Procedures and Results
      - MB.6.8.3.2 Reviews and Analyses of Simulation Cases, Procedures and Results

### 6.4 Software Review Process

- **MB.6.0** SOFTWARE REVIEW PROCESSES
  - MB.6.1 Purpose of Software Review
  - MB.6.2 Overview of Software Review Process Activities
  - MB.6.3 Software Reviews and Analyses
    - MB.6.3.1 Reviews and Analyses of the High-Level Requirements
    - MB.6.3.2 Reviews and Analyses of the Low-Level Requirements
    - MB.6.3.3 Reviews and Analyses of the Software Architecture
    - MB.6.3.4 Reviews and Analyses of the Source Code
    - MB.6.3.5 Reviews and Analyses of the Outputs of the Integration Process
  - MB.6.4 Software Testing
  - MB.6.5 Software Review Process Traceability
  - MB.6.6 Verification of Parameter Data Items
  - MB.6.7 Model Coverage Analysis for Design Models
    - MB.6.7.1 Model Coverage Analysis Criteria
    - MB.6.7.2 Model Coverage Analysis Resolution
  - MB.6.8 Model Simulation
    - MB.6.8.1 Model Simulation for Verification of the Model
    - MB.6.8.2 Model Simulation for Verification of the Executable Object Code
    - MB.6.8.3 Simulation Cases, Procedures and Results
      - MB.6.8.3.1 Development of Simulation Cases, Procedures and Results
      - MB.6.8.3.2 Reviews and Analyses of Simulation Cases, Procedures and Results

### 6.5 Software Testing

- **MB.6.0** SOFTWARE TESTING PROCESSES
  - MB.6.1 Purpose of Software Testing
  - MB.6.2 Overview of Software Testing Process Activities
  - MB.6.3 Software Testing and Analyses
    - MB.6.3.1 Testing and Analyses of the High-Level Requirements
    - MB.6.3.2 Testing and Analyses of the Low-Level Requirements
    - MB.6.3.3 Testing and Analyses of the Software Architecture
    - MB.6.3.4 Testing and Analyses of the Source Code
    - MB.6.3.5 Testing and Analyses of the Outputs of the Integration Process
  - MB.6.4 Software Review Process Traceability
  - MB.6.5 Verification of Parameter Data Items
  - MB.6.6 Model Coverage Analysis for Design Models
    - MB.6.6.1 Model Coverage Analysis Criteria
    - MB.6.6.2 Model Coverage Analysis Resolution
  - MB.6.7 Model Simulation
    - MB.6.7.1 Model Simulation for Verification of the Model
    - MB.6.7.2 Model Simulation for Verification of the Executable Object Code
    - MB.6.7.3 Simulation Cases, Procedures and Results
      - MB.6.7.3.1 Development of Simulation Cases, Procedures and Results
      - MB.6.7.3.2 Reviews and Analyses of Simulation Cases, Procedures and Results
How does the Supplement Link to the Core DO-178C

- The Supplement text refers to the core 178C processes wherever this is needed
- The Annex Table have links to both the supplement objectives and core 178C objectives

<table>
<thead>
<tr>
<th>Table MB.A-2 Objective 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

**CORE**
MBDV Supplement links to other Supplements

- There are supplements that cover Object Oriented design and Formal Methods
- It is possible that the applicant will use more than one supplement
- MB.4.0 Software Planning explains that the impact of all the supplements must be synthesized into a single approach and placed in the PSAC
- In some cases, the use of multiple approaches with the same PSAC may show a conflict between objectives. In this situation, the applicant must devise an approach and justify this in the PSAC.
Linking Supplements: Planning

MB.4.0 suggests consolidating a list to cover each objective with a clear statement of compliance for each objective.

One approach is to consolidate all objectives of DO-178C and the applicable supplements, and for each objective provide a statement of how compliance will be achieved, along with identifying any applicable life cycle data items.
When is the MBDV Supplement Mandatory

There has been an issue with past practices as they relate to models

- One major issue is who owns the model diagrams; many times these diagrams come from systems engineers.

- Sometimes engineers who don’t want to be bothered with “software stuff”
When is the MBDV Supplement Mandatory

► Past Practice
  - It has been much easier to create solutions in the form of a model and just pass the picture to the software guys to do the 178 stuff
  - Then the software guys just press the button, make code, confirm the primitive elements, and ship it!

► This is a flawed process which violates the spirit of 178B & 178C
When is the MBDV Supplement Mandatory

- The MBDV Supplement is mandatory whenever models are used for requirements.
- Regardless of who generates the models, the supplement applies.
- The supplement applies to requirements above the model as outlined in the Planning section MB.4.0.

Specifically related to model-based development, it may be difficult to separate system and software life cycle data. The Specification Model or the Design Model may originate from the system requirements and design process. Regardless of where these models originate, the guidance of this supplement applies and the objectives of this supplement should be satisfied.
Does It Always Apply?

What if only a “little” modeling is used, does the supplement apply? There's no auto-coding, can’t I skip the supplement?

► The supplement applies whenever modeling is used, regardless of the extent

► FAQ#4 highlights the rationale for this:

When a model is used to represent requirements, then the objectives, activities such as model coverage analysis, and software data items such as Software Model Standards are important regardless of how the Source Code is generated. The planning process should address each activity and objective as it relates to model usage on a particular project.
Addressing Unintended Function

- One of the major drawbacks to models is their ability to hide unintended function.
- This is not true in simplistic models but is a major issue with poorly structured, overly complex models.
- Use of good design standards, trace data tags, comments improves reviewability.
- Use of Model Coverage Analysis and Model Simulation helps assure correctness.
Major Section Review

- Design & Specification Model
- Standards
- Model Coverage Analysis
- Simulation
  - To verify requirements
  - To verify executable object code
Let’s Start at the Beginning

► After 20+ slides, it’s time to get specific
► Section MB.1 contains a wealth of information and should be read and understood
► It contains no guidance but it’s great reading in order to really understand everything else
Section MB.1

MB.1.0 is simple and straightforward

A model is an abstract representation of a set of software aspects of a system that may be used to support the software development process or the software verification process. The use of models may bring the benefits and capabilities of:

- Providing unambiguous expression of requirements and architecture.
- Supporting the use of automated code generation.
- Supporting the use of automated test generation.
- Supporting the use of analysis tools for verification of requirements and architecture.
- Supporting the use of simulation for partial verification of requirements, architecture, and/or Executable Object Code.
Section MB.1

MB.1.1 Purpose emphasizes that the document covers:

- Requirements as models (regardless of source)
- Any artifacts expressed as a model
- Any verification evidence derived from models

This document identifies the additions, modifications, and substitutions to DO-178C objectives when model-based development and verification are used as part of the software life cycle, and the additional guidance required. This includes the artifacts that would be expressed using models and the verification evidence that could be derived from them. Models developed in the system process that are used to define software requirements or software architecture are therefore applicable to this document.
Section MB.1.6

- Section MB.1.6 and its sub-section really provide significant insight into the rest of the document

- Important concepts are introduced
  - Characteristics of MBDV
  - Requirements from which the Model is Developed
  - Specification Model
  - Design Model
Two fully loaded sentences

- Requirement above the model
- External to the model
- Complete requirement & constraints for the model

For any type of model, there is a need to identify the requirements from which the model is developed. Those requirements should be external to the model itself and should be a complete set of requirements and set of constraints.
Section MB.1.6.2

The concepts of the Design Model and the Specification Model are introduced

Simply
- Specification Model = High Level Requirement (HLR)
- Design Model = Low Level Requirements (LLR)

The value of reading and re-reading MB.1.6.2 cannot be over emphasized

Section MB.1.6.3 and the rest of the document will not be clear without understanding MB.1.6.2
MB.1.6.2 Spec Model

A Specification Model is an abstract representation of externally observable properties of a software component, such as interface, functional, performance or safety characteristics. The Specification Model should express these characteristics unambiguously to support an understanding of the software functionality. It should only contain detail that contributes to this understanding and does not prescribe a specific software implementation or architecture except for exceptional cases of justified design constraints. Specification models do not define software design details such as internal data structures, internal data flow or internal control flow. Therefore a Specification Model may express high-level requirements but neither low-level requirements nor software architecture.

Glossary Definition

Specification Model - A model of externally observable properties of a software component representing high-level requirements, such as functional, performance, interface, or safety characteristics.
MB.1.6.2 Design Model

A Design Model prescribes software component internal data structures, data flow and/or control flow. A Design Model includes low-level requirements and/or architecture. In particular, when a model expresses software design data, regardless of other content, it should be classified as a Design Model. This includes models used to produce code.

Glossary Definition

**Design Model** - A model that defines any software design such as low-level requirements, software architecture, algorithms, component internal data structures, data flow and/or control flow. A model used to generate Source Code is a Design Model.
Important relationships between HLR and LLR

A single model cannot be both the HLR & LLR

A model cannot be classified as both a Specification Model and a Design Model.

For the purpose of this supplement, the term ‘high-level requirement’ refers to either:

- Any requirement contained in a Specification Model.
- Any requirement from which a Design Model is developed.

For the purpose of this supplement, the term ‘low-level requirement’ refers to:

- Any requirement contained in a Design Model.
Table MB.1-1 and DP#1

- Table MB.1-1 in section MB.1.6.3 has examples of the various possible relationships between models and life-cycle data (not exhaustive).

- Appendix B has a Discussion Paper (DP#1) that further explains models usage examples in pictorial format.

- DP#1 links the examples back to MB.1-1.
### Table MB.1-1

The notes that go along with this table are very important as they make some subtle points very clear.

<table>
<thead>
<tr>
<th>Process that generates the life-cycle data</th>
<th>MB Example 1</th>
<th>MB Example 2</th>
<th>MB Example 3</th>
<th>MB Example 4 <em>(See Note 1)</em></th>
<th>MB Example 5 <em>(See Note 1)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>System Requirement and System Design Processes</td>
<td>Requirements allocated to software</td>
<td>Requirements from which the Model is developed</td>
<td>Requirements from which the Model is developed</td>
<td>Requirements from which the Model is developed</td>
<td>Design Model</td>
</tr>
<tr>
<td>Software Requirement and Software Design Processes</td>
<td>Requirements from which the Model is developed</td>
<td>Specification Model <em>(See Note 2)</em></td>
<td>Specification Model</td>
<td>Design Model</td>
<td>Textual description <em>(See Note 3)</em></td>
</tr>
</tbody>
</table>
Separate Models for HLR & LLR – *Example C*

- Shows the case where some HLR are model and some HLR are text
- All LLR is in a model
- Remember that real situations will not be 100% model-based
A Design Model coming from Systems - Example E

► This is a great example that can happen quite often and will lead to a lot of debate

► Luckily DP#1 helps clarify this situation
Example E

► Remember that the Design Model is LLR
► We must have also HLR somewhere
► If the Design Model came from systems then the HLR must live in the systems process AND the supplement applies there

This example shows the case where System Requirements allocated to Software are provided as a Design Model, as described in section MB.2.1.1. This is similar to Example 5 in Table MB.1-1 of this supplement, where the requirements from which the Design Model is developed are created by the system processes.

Note: In that case, objectives for HLR apply to the requirements from which the Design Model is developed. Those requirements are part of the SRATS and this supplement only applies to the part of the SRATS from which the Design Model was developed.
Model Coverage & Simulation

The two biggest new concepts in the document are Model Coverage and Simulation.

Model Coverage is an activity covered in section MB.6.7.

Model Simulation is a powerful advantage of MBDV covered in section MB.6.8.
Section MB.6.7 Model Coverage

Model Coverage Analysis is a technique to show that the requirements within the Design Model are covered by testing.

Model coverage analysis determines which requirements expressed by the Design Model were not exercised by verification based on the requirements from which the Design Model was developed. This analysis contributes to the detection of unintended functions in the Design Model and potentially in the final Executable Object Code.
As can be seen below, Model Coverage can be used to show compliance of LLR. Note that HLR coverage is assumed to be correct otherwise this step will fail.

Model Coverage Analysis—An analysis that determines which requirements expressed by the model were not exercised by verification based on the requirements from which the model was developed. The purpose of this analysis is to support the detection of unintended function in the Design Model, in a context where high-level requirements coverage has been achieved.

| 1 | Low-level requirements comply with high-level requirements. | MB.6.3.2.a | MB.6.3.2 |  ● |  ● | ○ | Software Verification Results | MB.11.14 |  ● |  ● |  ● |
|---|---|---|---|---|---|---|---|---|---|

Table MB.A-4
Section 6.8 Model Simulation

- Simulation can be used in a limited fashion to support some aspects of verification
- Strict rules are placed on the use of simulation

Model Simulation

The goal of the MB.6.8 subsections is to provide guidance, including limitations, to allow the use of simulation without compromising the rigor appropriate for the software level. The use of model simulation in compliance with this supplement does not alleviate the guidance of DO-178C sections 6.3 and 6.4. When simulation is used, all objectives of high-level requirements verification, software design verification, Executable Object Code verification, and software testing verification, should still be satisfied (see MB.6.3.1, MB.6.3.2, MB.6.3.3, and MB.6.4). Model simulation may support the achievement of some of these objectives. In
Model Simulation for Verification of the Model

- This section can be used for both Design and Spec Models
- It can be used to support the following:

  - Compliance to system requirements for Specification Models (see MB.6.3.1.a).
  - Compliance to software high-level requirements for Design Models containing low-level requirements (see MB.6.3.2.a).
  - Compliance to software high-level requirements for Design Models containing software architecture (see MB.6.3.3.a).
MB.6.8.2

Model Simulation for the Verification of Executable Object Code

- This section is the most restrictive to assure that the techniques are not misused.
- With considerable planning and demonstration it may be possible to provide limited assessment of executable object code behavior within the simulation environment.

Verification of the Executable Object Code is primarily performed by testing. This can be partially assisted by a combination of model simulation and specific analyses as described below. This combination can be used to partially satisfy the following software testing and test coverage objectives:
Simulation Cases, Procedures & Results

- If you don’t run cases & procedures then the model is just static & boring
- Simulation, via defined cases and procedures, provides a structured methods to assess the model correctness and completeness
- These cases and procedures may also be used on the target software/hardware for test
New Objective on Simulation

The combination of objectives and activities related to simulation are included in the Annex Tables MB.A-4, MB.A-6 and MB.A-7

Table MB.A-4

<table>
<thead>
<tr>
<th>MB 14</th>
<th>MB.6.8.3.2.a</th>
<th>MB.6.8.1 MB.6.8.3.2</th>
<th>MB.11.14</th>
<th>Software Verification Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB 15</td>
<td>MB.6.8.3.2.b</td>
<td>MB.6.8.1 MB.6.8.3.2</td>
<td>MB.11.14</td>
<td>Software Verification Results</td>
</tr>
<tr>
<td>MB 16</td>
<td>MB.6.8.3.2.c</td>
<td>MB.6.8.1 MB.6.8.3.2</td>
<td>MB.11.14</td>
<td>Software Verification Results</td>
</tr>
</tbody>
</table>
Model Standards - MB.11.23

- There is a new section describing the attributes of the required Model Standard

Aspects include:
- Methods and tools
- Syntax and limitations of feature use
- Design constraints
- Delimiting and tagging/trace data
- Method to provide derived items to the safety process
- Comment syntax
Trace Data

► Each element in a model is there for a reason
► Make sure that you can trace each box, each constant value, each construct to a requirement and then to the code
► Models are requirement/architecture but may also represent design
► Poorly tagged and poorly traced models will make a compliance finding a nightmare
► “Leap of Faith” decomposition should be avoided
► Pick a methods and define it in the plans and standards
Test Environments

► The testing environment may have a one or more models

► These models are considered part of the test environment and are not directly covered by 178C or the supplement

► FAQ #9 however does provide a vehicle to have a common approach to test environment models

Section MB.11.3 items c and d provide more details on the treatment of models in the Software Verification Plan. Section MB.11.15 related to configuration management should cover both the tools and the model version used in the environment.

In summary, the models used in a test environment should be planned, identified, assessed, and controlled just like any other aspect of the test environment.
Planning for Models

► If you use more than one model then each instance of the model means an instance of the MBDV Supplement must be assessed

► Think of your planning activity as a big excel sheet

<table>
<thead>
<tr>
<th>Objective #</th>
<th>Text HLR Requirements</th>
<th>Text LLR Requirements</th>
<th>Model #1 HLR</th>
<th>Model #2 LLR</th>
<th>Model #3 LLR</th>
<th>Formal Model #1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-1 Obj 1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>A-1 Obj 2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>A-1 Obj 3</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>A-1 Obj 4</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MB.A-1 Obj 1</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MB.A-1 Obj 2</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MB.A-1 Obj 3</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MB.A-1 Obj 4</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>FM.A-1 Obj 1</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>FM.A-1 Obj 2</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>FM.A-1 Obj 3</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>FM.A-1 Obj 4</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Things to Remember

- Models hide complexity
  - This can hide unintended function
  - Toolsets have setup files and libraries that need control
  - Models are ambiguous without model standards
  - Models must have requirements above them
  - Models cannot be verified against themselves

- We are obliged to assure that the models perform their intended functions and behaviors
  - This is much more than verifying the operation of primitive elements
Committee Status

► The RTCA/EUROCAE committees for 178C issues are not accepting new member access to the websites.

► Public comment will be through the RTCA Final Review and Comment (FRAC) process for each document.

► Final approval is expected by year’s end.
Thank You

Mark Lillis
Goodrich Pump & Engine Control Systems
West Hartford, Connecticut
mark.lillis@goodrich.com

Questions?