Complying with DO-178C and DO-331 using Model-Based Design

Bill Potter
MathWorks, Inc.

ABSTRACT

This paper addresses how recently published revisions of aircraft certification documents affect developers using Model-Based Design. It focuses on SAE ARP4754A and RTCA DO-178C, in particular RTCA DO-331, the Model-Based Development and Verification Supplement to DO-178C and DO-278A. The previous version of the software standard, RTCA DO-178B, lacked guidance on modern development and verification practices such as Model-Based Design, object-oriented technologies, and formal methods. The core DO-178C document is a relatively minor update to the previous DO-178B standard, but the technology supplements nevertheless provide new guidance in how to adapt these modern technologies on a DO-178C project. The supplements may add, delete, or modify objectives in the core DO-178C standard for the applicable technology. Conversely, if the technology does not affect an existing DO-178C objective, then the objective is used as is.

ARP4754A addresses the complete aircraft development cycle, from systems requirements through systems verification. It covers requirements, integration, and verification for three levels of abstraction: aircraft, systems, and item. An item is defined as a hardware or software element having bounded and well-defined interfaces. According to the standard, aircraft requirements are allocated to system requirements, which are then allocated to item requirements.

DO-178C addresses the complete software development cycle, from software requirements to software verification. The system processes and software processes are related, in that system requirements are allocated to software requirements. Both ARP4754A and DO-178C address this allocation step. When models are used as part of the software development life cycle, even if these models are developed in the systems processes, DO-331 becomes applicable to the development and verification of those models.

With Model-Based Design, engineers develop and simulate system models comprising hardware and software using block diagrams and state charts. They then automatically generate, deploy, and verify code on their embedded systems. This approach lets system, software, and hardware engineers collaborate using the same tools and environment to develop, implement, and verify systems. Developers of complex systems need to leverage this technology and be mindful of the governing standards such as ARP4754A, DO-178C, and DO-331.

INTRODUCTION

This article highlights the recently updated commercial aviation standards and how they affect developers using Model-Based Design. ARP4754A and DO-178C are the focus, especially DO-331, the Model-Based Development and Verification Supplement to DO-178C.

THE EXISTING STANDARDS

Few industries place more importance on verification or prescribe more process guidance than commercial aviation. The FAA and its European equivalent, EASA, provide guidance using standards such as ARP4754 for aircraft systems and DO-178B for flight software. Other standards, such as DO-254 for hardware and DO-278 for ground or space-based software, are also used.
These standards, however, are more than a decade old and are showing their age. For example, they lack guidance on modern development and verification practices such as Model-Based Design, object-oriented technologies, and formal methods. This led FAA and EASA to work with aircraft manufacturers, suppliers, and tool vendors including MathWorks to update standards based on modern technologies. Rather than significantly modify the standards, they created technology supplement documents.

Table 1 – Aircraft certification documents and recent updates

<table>
<thead>
<tr>
<th>Document</th>
<th>Release</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAE ARP4754A</td>
<td>12/01/2010</td>
<td>Aircraft Systems</td>
</tr>
<tr>
<td>RTCA DO-178C</td>
<td>12/13/2011</td>
<td>Airborne Software</td>
</tr>
<tr>
<td>RTCA DO-254</td>
<td>04/19/2000</td>
<td>Airborne Electronic Hardware</td>
</tr>
<tr>
<td>RTCA DO-278A</td>
<td>12/13/2011</td>
<td>CNS/ATM Software</td>
</tr>
<tr>
<td>RTCA DO-330</td>
<td>12/13/2011</td>
<td>Software Tool Qualification</td>
</tr>
<tr>
<td>RTCA DO-331</td>
<td>12/13/2011</td>
<td>Model-Based Development and Verification Supplement</td>
</tr>
<tr>
<td>RTCA DO-332</td>
<td>12/13/2011</td>
<td>Object-Oriented Technology Supplement</td>
</tr>
<tr>
<td>RTCA DO-333</td>
<td>12/13/2011</td>
<td>Formal Methods Supplement</td>
</tr>
</tbody>
</table>

TRANSITIONING TO NEW STANDARDS

ARP4754A

ARP4754A addresses the complete aircraft development cycle from requirements through verification. It discusses requirements, integration, and verification for three levels of abstraction: aircraft, systems, and item. An item is defined as a hardware or software element having bounded and well-defined interfaces. According to the standard, aircraft requirements are allocated to system requirements, which are then allocated to item requirements.

This last point in ARP4754A, addressing the allocation of system requirements to hardware and software components, is significant to aircraft manufacturers and their suppliers. In the past, some suppliers may have claimed that subsystem development was beyond the scope of ARP4754, even for complex subsystems containing hardware and software, but not anymore. ARP4754A also more clearly refers to DO-178 and DO-254 for item design. In fact, the introductory notes in ARP4754A acknowledge that its working groups coordinated with RTCA Special Committee 205 (SC-205) / EUROCAE WG-71 to ensure that the terminology and approach being used are consistent with those being developed for the update to DO-178B / ED-12B for consistency with ARP4754A.

The FAA issued Advisory Circular 20-174 in September 2011, formally recognizing ARP4754A as an acceptable method of establishing a development assurance process.

DO-178C

Not surprisingly, one of the first new changes in DO-178C is an explicit mention of ARP4754A in Section 2 System Aspects Relating to Software Development, which serves to further align the standards.

This section discusses those aspects of the system life cycle processes necessary to understand the software life cycle processes. System life cycle processes can be found in other industry documents (for example, SAE ARP4754A).
Clarification updates like this one aside, DO-178C does not differ significantly from DO-178B, at least at first glance. In fact, a casual reader might miss an item mentioned in Section 1.4: How to Use this Document.

One or more supplements to this document exist and extend the guidance in this document to a specific technique... if a supplement exists for a specific technique, the supplement should be used to add, delete, or otherwise modify objectives, activities, explanatory text, and software life cycle data in this document to address that technique, as defined appropriately in each supplement.

In other words, the big changes are captured in the supplemental documents, specifically those listed in Table 1, such as RTCA DO-331, Model-Based Development and Verification Supplement to DO-178C and DO-278A.

It should be noted that neither the FAA nor EASA have formally recognized DO-178C/ED-12 at this time. These authorities are expected to accept these documents in late 2012 or early 2013.

INTRODUCTION TO MODEL-BASED DESIGN

With Model-Based Design, engineers develop and simulate system models composed of hardware and software using block diagrams and state charts as shown in Figures 1 and 2. They then automatically generate, deploy, and verify code on their embedded systems. With MATLAB® and Simulink® from MathWorks, engineers generate C, C++, Verilog, and VHDL code for implementation on MCU, DSP, FPGA, and ASIC hardware. This lets system, software, and hardware engineers collaborate using the same tools and environment to develop, implement, and verify systems. Developers of complex systems may leverage all the technologies listed above and be mindful of the governing standards such as those listed in Table 1.

![Figure 1 – Autopilot system model with controller and plant](image1.png)

A key benefit of Model-Based Design is early verification. Verification starts as soon as models are created and simulated using tests based on high-level requirements. This verification can be accomplished on a desktop computer, without the need for the target hardware or software.

![Figure 2 – Autopilot controller model for software](image2.png)
TRANSITIONING TO NEW STANDARDS USING MODEL-BASED DESIGN

ARP4754A

ARP4754A recommends the use of modeling and simulation for several integral process activities, including requirements capture and requirements validation.

ARP4754A Table 6 recommends analysis, modeling and simulation (tests) for validating requirements at the highest Development Assurance Levels (A and B). For Level C, modeling is listed as one of several recommendations. While ARP4754 made similar recommendations, ARP4754A provides more insight and states that an accurate environment model, such as the plant model shown in Figure 1, is an essential part of a system model.

Model use for requirements validation typically uses a model of the environment of a system being developed, which is interfaced to a prototype of a design solution for those requirements. An environment model that is representative of the environment of the system being developed provides a high degree of functional coverage in exercising either a simulated or real system.

Also new with ARP4754A is that a graphical representation or model can be used to capture system requirements. It also notes that a model can be reused later for software and hardware design.

Models used to capture requirements and then directly used to produce embedded code (Software or HDL) come within the scope of DO-178B/ED-128 and DO-254/ED-80 from the time that certification credit is to be taken until the software or hardware is returned to the system processes for system verification.

If you do use models to capture requirements, ARP4754A recommends you consider the following:

1. Identify the use of models/modeling
2. Identify the intended tools and their usage during development
3. Define modeling standards and libraries

DO-178C AND DO-331

A long standing issue with DO-178B for practitioners of Model-Based Design is the uncertainty in mapping DO-178B objectives to Model-Based Design artifacts. Addressing this mapping was a main goal of the DO-178C subgroup (SG-4) focused on Model-Based Design. No single mapping sufficed, so several mappings are provided in DO-331. Some include the concept of a specification model but that approach is not typically employed by Simulink users. The other concept of a design model is a more natural mapping for Simulink users. It is important to note that a single model cannot be classified as both a specification model and a design model, and this is explained in detail within DO-331.

Table 3 shows popular mappings of life-cycle data to model usage based on DO-331, Section MB.1.6.3.

<table>
<thead>
<tr>
<th>Process that generates life-cycle data</th>
<th>Model-Based Design Example 1</th>
<th>Model-Based Design Example 4</th>
<th>Model-Based Design Example 5</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>
</tr>
</tbody>
</table>

Table 3 - Example mappings of life-cycle data to model usage
The essence of this table is the following:

1. A model can be used for design (of the system, the software, or both) and should be developed using requirements external to the model (e.g., a textual document or requirement database).
2. Source code can be generated directly from the design model (by hand or automatically).

Of course, with 125 pages, DO-331 has a lot more to offer than that, but this table ties together the basic concepts of Model-Based Design for system and software development. Example 5 does this particularly well in that it shows that a model used initially for system design is elaborated and reused for software design and code generation. It should be noted that DO-331 applies to models, even when code is developed manually from the model without the use of an automatic code generator.

For example, the controller shown as a component in the system model in Figure 1, and by itself as a software model in Figure 2, is used during system design, reused as an entry point for software design, elaborated during detailed software design (for example by discretizing continuous time blocks and changing double-precision data to single-precision), and then used as input for embedded code generation. The test cases used for system requirement validation likewise are reused on the model, source code, and executable object code to perform functional testing and collect coverage metrics.

Engineers may choose any of the mappings specified in DO-178C. However, the approach in Example 5 of using and reusing models for systems and software design along with code generation has provided Simulink and Embedded Coder users significant benefits, including reduced cost and improved quality. Many engineers will appreciate that this same approach is now clearly acknowledged as an acceptable means to certification by the guidance offered in the governing standards.

DO-331 introduces two important techniques, simulation and model coverage analysis, that may be used with Model-Based Design to satisfy objectives for design models. DO-178C calls out reviews, analyses, and testing as appropriate verification techniques for requirements, design, source code, and executable object code. DO-331 adds simulation as a valid technique to verify models, and this specific type of analysis capability is available when using modeling tools. For some, but not all, objectives for high and low-level requirements, DO-331 allows the use of simulation to satisfy objectives in place of traditional reviews and analysis. In fact, simulation can be more effective than reviews in determining correctness, because simulation provides a means of predicting the dynamic behavior of a system. But use of simulation does require some extra effort, for example the simulation cases must be treated similarly to test cases in that they need to be reviewed against the higher level requirements from which the model was developed.

Model coverage analysis is also a new technique called out in DO-331, and in fact this analysis is a required activity when using design models. One of the concerns raised by the certification authorities during the development of DO-178C and DO-331 was that past projects using Model-Based Design sometimes had vague requirements associated with very complex models. This raised the concern that unintended functionality could be introduced into the system during the model development process. Model coverage analysis is performed during model simulation and the simulation cases must be based on the requirements from which the model is developed. This is similar to how code coverage is performed; it is done using test cases based on the software requirements, not on the code itself. DO-331 contains examples of the types of model coverage metrics that should be considered for the analysis. Some commercial modeling products, such as Simulink Verification and Validation, provide model coverage tools that can be used for this activity.
TOOL QUALIFICATION CONSIDERATIONS

The factors used to determine if a tool needs to be qualified have not changed from DO-178B to DO-178C. Section 12.2.1 of DO-178C states: Qualification of a tool is needed when processes of this document are eliminated, reduced or automated by the use of a software tool without its output being verified as specified in Section 6. For example, a modeling tool does not need to be qualified as long as the output of the tool (the model) is verified per DO-331 model verification objectives. If an automatic code generator is used, and the generated code (tool output) is reviewed then qualification is not necessary. If, however, the code review is to be eliminated then the code generator must be qualified.

With Model-Based Design there are numerous tools available to perform model checking and model verification. If these tools are used for credit, then they must be qualified. For example, a model coverage tool used to perform model coverage analysis would need to be qualified just as code coverage tools need to be qualified. Tool vendors, including MathWorks, provide tool qualification kits for those tools that can be used for certification credit. Examples of qualifiable tools from MathWorks for both model verification and code verification are:

- Simulink Report Generator™
- Simulink Verification and Validation™ (Model Advisor and Model Coverage)
- Simulink Code Inspector
- Polyspace® products
- SystemTest™

OBJECT-ORIENTED TECHNOLOGY AND FORMAL METHODS

With Model-Based Design it is possible to also use object-oriented technology and formal methods. For example, an automatic code generator could be used to generate C++ code, or a formal methods tool such as Simulink Design Verifier™ could be used to prove properties within a model. Additionally, the Polyspace formal methods tools could be used to find vulnerabilities in C++ code or to detect certain run-time errors in C or C++ code. In this case DO-332 and DO-333 may also apply. It is not the intent of this paper to detail the use of those technologies, but it is important to note they can be used concurrently.

SUMMARY/CONCLUSIONS

Since the original publication of DO-178B in 1992, the use of modeling and simulation has become the norm for many commercial and military projects. In fact, many systems built using these technologies have been certified under DO-178B. Unfortunately, engineering organizations and the certification authorities have been somewhat inconsistent with the application of this technology and the approval of systems developed using it. With the release of ARP4754A, DO-178C and DO-331, there is now guidance on how to use modeling and simulation in a robust and effective manner that both software practitioners and certification authorities can understand.

REFERENCES

1. SAE ARP4754A Guidelines for Development of Civil Aircraft and Systems, dated December 2010
2. RTCA DO-178B Software Considerations in Airborne Systems and Equipment Certification, dated December 1, 1992
3. RTCA DO-278 Guidelines for Communication, Navigation, Surveillance and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance, dated March 5, 2002
4. RTCA DO-254 Design Assurance Guidance for Airborne Electronic Hardware, dated April 19, 2000
5. RTCA DO-178C Software Considerations in Airborne Systems and Equipment Certification, dated December 13, 2011
7. RTCA DO-330 Software Tool Qualification Considerations, dated December 13, 2011
8. RTCA DO-331 Model-Based Development and Verification Supplement to DO-178C and DO-278A, dated December 13, 2011
CONTACT INFORMATION

Bill Potter
MathWorks
Email: bill.potter@mathworks.com
Phone: 623-433-8801

ACKNOWLEDGMENTS

DEFINITIONS/ABBREVIATIONS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAA</td>
<td>Federal Aviation Administration</td>
</tr>
<tr>
<td>EASA</td>
<td>European Aviation Safety Agency</td>
</tr>
<tr>
<td>MCU</td>
<td>Micro Controller Unit</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processor</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuit</td>
</tr>
<tr>
<td>HDL</td>
<td>Hardware Description Language</td>
</tr>
</tbody>
</table>

APPENDIX