Transitioning to RTCA DO-178C/DO-278A:
A Business Manager Brief
Hoyt Lougee, Engineering Director - Foliage, Inc.

ABSTRACT
With the release of the new development guidance for certifiable aviation software, RTCA DO-178C/DO-278A, executives and business managers for manufacturers of both airborne and ground-based systems are examining the short- and long-term impacts on their certifiable-product development approaches with respect to cost, schedule, and risk. Although the changes within DO-178C/DO-278A proper are relatively few, manufacturers can expect critical wide-ranging business impacts on their current certifiable aviation software development. More significant are the four detailed supplements intended to address 20 years of technology and process progress since the last major revisions of development guidelines.

The new guidance represents a significant change in the FAA's posture towards regulated software development with DO-178C/DO-278A tightening some previously established controls, while also establishing concrete guidance for greater flexibility in development approaches. This flexibility, however, must be carefully examined, weighing potential business costs and benefits, to establish the most efficient certifiable product-development approach. This white paper discusses these shifts from the business perspective, providing business management visibility into the emerging challenges and opportunities associated with the updated guidance.

INTRODUCTION
The FAA released DO-178B, Software Considerations in Airborne Systems and Equipment Certification, 20 years ago to provide guidance on the development of FAA certifiable airborne software¹. Since then, the software development world has matured significantly, with advances in technologies and processes providing new opportunities for manufacturers to streamline development and tackle

¹ Note that DO-278, Guidelines for Communications, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance, was released 10 years later, tightly tied to the content of DO-178.
increasingly complex systems. For example, today, state-of-the-art software development often involves the use of advanced model-driven development with the corresponding benefit of automated code generation. Prior to the release of DO-178C, the FAA often scrutinized closely the certification of a product developed in this manner, requiring additional meetings, white papers, and presentations, which added—from the business planning perspective—significant risk. Indeed, more often than not, the introduction of model-driven development resulted in confusion, stalled product completion, and delayed time to market.

Much of this progress, therefore, has been problematic in the certifiable-software development domain. The general guidance provided by DO-178B/DO-278 often served only to muddle matters further—with disagreements on the acceptability and treatment of emerging technologies and processes among the various aircraft component manufacturers, individuals within the FAA Aircraft Certification Offices (ACOs), and their Designated Engineering Representatives (DERs).

The result: industry has been hobbled in its ability to effectively plan and execute certifiable-software development, and has not kept pace with the state of the art in software engineering at large. In fact, the only dependable attribute of certifiable-software development seems to be the consistency of missed schedules and burst budgets, at times attributed to the shifting interpretations and dated guidance. The benefits of new tools, techniques, and processes—as leveraged successfully in non-certifiable-software development—have been at best elusive, at worse unavailable, as the certifiability overhead has taken its toll.

In 2001, to address common industry and certification-authority questions, RTCA released DO-248B, Final Report for Clarification of 178B “Software Considerations in Airborne Systems and Equipment Certification.” This report addressed DO-178B errata, as well as provided more detail on specific issues in the form of FAQs. Still, interpretation remained problematic and continued to hinder the ability of manufacturers to plan with confidence the cost and schedule impact of the certification efforts. To fully address industry’s concerns and emerging technology, the FAA, in partnership with the aviation industry, set out to update the guidance in 2004. In December of 2011, updated guidance was released:

- RTCA DO-178C, Software Considerations in Airborne Systems and Equipment Certification
- RTCA DO-278A, Guidelines for Communications, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance
In addition, supplements were released to provide a detailed treatment of four critical areas:

1. RTCA DO-330  
   *Software Tool Qualification Considerations*

2. RTCA DO-331  
   *Model-Based Development and Verification Supplement to DO-178C and DO-278A*

3. RTCA DO-332  
   *Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278*

4. RTCA DO-333  
   *Formal Methods Supplement to DO-178C and DO-278*

This guidance offers a more effective roadmap for certifiable-software development, lowering risks and providing a dependable basis for cost/schedule projection. The initial path forward, however, will not be easy. The guidance must be carefully studied, and, for the specifics of each certifiable product manufacturer, its impacts assessed and detailed plans developed. The certification authorities should then approve these plans prior to the start of the development effort. Only then can manufacturers be confident that the new guidance will work for them, enabling advances in technology and process to yield cost, schedule, and risk-mitigation benefits; while preventing costly misinterpretations and 11th-hour roadblocks.

**WHAT HAS CHANGED?**

In the big picture, two types of changes have occurred:

1. Clarification of original guidance
2. Detailed guidance related to major shifts in technology/process

Both areas will have significant impact on the way certifiable-software development is managed.

**CLARIFICATIONS**

Ironically, a major clarification within DO-178C/DO-278A doesn’t concern the software development process directly. Instead, the clarification relates to the certifiable-software development interaction with the higher-level system processes, as detailed in the newly revised ARP4754, *Guidelines for Development of Civil Aircraft and Systems*. The potential impact of these clarifications must not be underestimated. The certification authorities are looking to tighten up this interaction with formal, iterative, closed-loop data flow among the system, hardware, and software processes. Across the aviation industry, addressing the information flow between systems and software, especially with respect to interfaces, requirements allocation, and safety-related requirements has varied. The
information flow has ranged from a cursory, after-the-fact treatment to a very formal, detailed multi-stage analysis—including elaborate System Safety Analysis, Functional Hazard Analysis, Functional Failure Path Analysis, and Failure Modes and Effects Analysis. A manufacturer’s understanding of where its organization stands in this continuum, relative to the new guidance, will be critical in planning and risk management of its next certifiable-software development efforts. Very few organizations will emerge without having to modify their processes at all.

The updated guidance also provides direction on the oversight of suppliers, establishing formal expectations for auditable supplier compliance and oversight. Interestingly, the traditional and logical interpretation of DO-178B already allowed for certification-authority oversight into a supplier’s applicable certifiable-software development processes, so alignment to the revised DO-178C supplier-oversight processes should not be significant. Opening up the manufacturer-supplier relationship to detailed oversight planning with certification authority audits will, however, introduce the need for documented coordination and may drive additional geographically distributed supplier audits—first by the manufacturer and then by the certification authorities (and/or their designees).

Another significant change in the existing guidance is the introduction of a Parameter Data Item File, a data set that can influence the behavior of the software without modifying the executable object code. Formal guidance is now provided in the development, control, and verification of this data, which may, or may not, align well with a manufacturer’s current practices—and may have a significant impact on the planning, development, verification, and configuration management of the certifiable product.

The expectations for traceability have been clarified also. Traceability is the backbone of a certifiable-software development process, and, if performed or reworked after the fact, can be incredibly expensive and error prone. Although most organizations will have little impact from the refinement of the traceability guidance, a careful up-front review of the expectations will be necessary to confirm alignment.

To summarize, there is little doubt that the revisions to DO-178B/DO-278 will impact an organization’s certifiable development efforts—the only question is how (and how much). A careful and thoughtful gap analysis must be a priority, sooner rather than later, to understand and accommodate the impact—providing a solid basis on which to plan the next certifiable-software development effort.

SUPPLEMENTS
As discussed earlier, the revision to the guidance is not limited to clarification; also included are four new, complex addendums to address the state of the art and practice in software development.

1. RTCA DO-330, Software Tool Qualification Considerations
Advances in technologies are typically associated with promises of significant efficiency gains, driven by ever more prolific and complex tools. Tool guidance has migrated from DO-178B/DO-278 to a separate
detailed addendum, which describes a significantly more complex treatment of tools—driving the treatment of tools to more closely align with the treatment of certifiable software.

For example, DO-178B/DO-278 addressed the acceptability and qualification of tools within the certifiable-software development environment with a simple partition into Development tools and Verification tools. Development tools were to be developed with the same rigor as the certifiable software, whereas Verification tools only required basic testing against documented requirements. In contrast, DO-330 identifies five Tool Qualification Levels (TQLs) based on the tool use and its potential impact to the software life cycle process, with the level of tool qualification rigor varying according to the TQL.

Overall, manufacturers can expect the tool qualification burden to be higher, with additional expectations/formality across the “development” tool spectrum, and a slight additional bump on certain “verification” tools.

Establishing a process for certifiable-software development and selecting the associated tools can be a daunting task, with large costs and risks associated with both short- and long-term impacts. Therefore, with respect to the cost/benefit tradeoffs of the adoption of tools and tool suites to be used, each manufacturer should always first consider the impacts of the revised tool-qualification guidance with respect to both the long-term (multiple product development efforts), before addressing their short-term immediate project needs.

The associated tool selections/qualifications made will need to be detailed and justified in the planning documentation for each instantiation of a certifiable-software development effort. That said, the new guidance does address the reuse of previously qualified tools, with formalization of a common-sense adoption of existing qualification data for demonstrably similar applications (and TQLs). With this formalization and an understanding of the rigor required—tool strategies can and should be developed with more confidence.

As with the clarification impact discussed above, all certifiable-software development organizations will be affected by the new tool qualification guidance, with the size of the impact and the ease of adoption critically dependent on timely planning.
2. RTCA DO-331, Model-Based Development and Verification Supplement to DO-178C and DO-278A

DO-178C/DO-278A defines a model as an abstract representation of a set of software aspects of a system that is used to support the software development or software verification processes. Ultimately, this generally translates into the use of specific methods and tools in describing requirements, auto-generating code and tests, and analyzing/simulating requirements, architecture, and/or executable code.

Certifiable-software development is inherently expensive, especially at higher-criticality levels. Though comprised of best (albeit conservative) development practices, in general, putting together a formal auditable configuration can be an overwhelming task. As a result, many tools are available to reduce the development/verification burden through automation and analysis—some specific to certifiable-software development; some not.

In the certifiable software arena, industry has pushed hard for the acceptance of model-based development and a clear path to certifiability. With the promise of faster, more reliable development, many tool suppliers are fielding tools for a wide variety of purposes—each facing the formidable challenge of certification-authority acceptability. These tools typically are quite expensive, especially when considering the costly tool-qualification packages necessary for adoption into certifiable-software development.

Prior to the release of DO-178C, however, the guidance has been murky with respect to the model-based development qualification/certification hurdles that must be addressed. The introduction of DO-331, however, is a mixed blessing to the model-based development proponents. The new guidance clearly establishes the model-based development expectations, but these expectations include some very specific constraints, which enabled the retention of the original DO-178B/DO-278 breakdown between high-level requirements (the Specification Model), and low-level requirements and architecture (the Design Model).

One of the historic complaints of the certifiable-software development process was the complexity of the guidance, with its many nuances and a seemingly endless learning curve. The certification authorities, however, have been unmoved: only those capable of understanding the complex guidance established to provide flight-worthy confidence are allowed to play in the certifiable-software development game. The guidance associated with DO-331 is no exception. The guidance is complex, the concepts are not straightforward, and early adopters will face an uphill battle as the interpretations are wrung out.

Therefore, although model-based development can provide benefit, a full understanding of the guidance and expected impacts of initial adoption on a certifiable-software development program will
be necessary to truly understand the costs and benefits of migrating to model-based development. Note that the processes described, however, were not developed out of whole cloth. Many companies have already incorporated model-based development under the DO-178B/DO-278 guidance. Although these companies may have to modify their approaches to suit the new guidance; they have, in fact, laid a solid groundwork for model-based development. Even though adoption of model-based development is associated with challenges, it will quickly become a basic cost of playing in the certifiable product development arena—as manufacturers leverage the efficiency and associated competitive advantage that model-based development will bring.

In summary, model-based development can and will provide benefit to the certifiable product development community—with graphical building blocks and auto-code generation able to reduce the overall complexity of the software development. The costs and benefits, however, must be carefully analyzed and weighed. Business management must temper its expectations of an immediate return for the adoption of model-based development. Longer-term investment (in tools, processes, and training) will be required to fully benefit from the new technology and processes—and will be necessary to maintain competitive product costs and time-to-market.

3. **RTCA DO-332, Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A**

Object-oriented technology (OOT) is not new, with early representation stretching back to the 1960s and 1970s. Neither the release of DO-178B in 1992 nor the release of DO-278 in 2002 addressed the unique considerations and challenges represented by OOT and languages. Instead, the guidelines were loosely based on the more traditional and, at the time, more widely-used structured-design methodologies. In the 20 years after the release of DO-178B, OOT has overtaken structured design, from academic treatment and training to application in the development of complex and life-critical safety systems. In fact, OOT reigns supreme in nearly all domains, except, however, in certifiable airborne applications.

In late October 2004, after a handful of manufacturers fielded object-oriented airborne applications using the ill-fitting existing guidance, the FAA\(^2\) co-sponsored the Object-Oriented Technology in Aviation (OOTiA) project with the National Aeronautics and Space Administration (NASA) to address the challenges and constraints associated with the incorporation of OOT in a development process intended to meet the extreme rigors of FAA certifiable-software development. The resulting four

---

\( ^2 \) Handbook for Object-Oriented Technology in Aviation (OOTiA, Volume 1 Handbook Overview, Revision 0, October 26, 2004; Section 1.1.2 Background
volumes set the stage for the adoption of OOT, including ramifications on DO-178B/DO-278, best practices, and explicit guidance for certification authorities and their designees.

After seven years, however, the prevalence of object-oriented technology in aviation still doesn’t reflect the overall adoption in industry as a whole. With the promises of greater efficiency, the proliferation of object-oriented toolsets, and the growing population of engineers whose primary academic training was in OOT, the question of why object-oriented technology hasn’t taken hold is, indeed, enigmatic.

The answer lies in the difficulties inherent in meeting the extreme expectations for developing and verifying deterministic, verifiable software that forms the backbone of FAA guidance. Being able to address the challenges related to object-oriented complexities; such as hierarchical encapsulation, polymorphism, dynamic memory management, and virtualization; did not seem to be worth the effort in an already risk-rich certifiable-product development context.

The new guidance, DO-332, supersedes the OOTiA, and provides a more concise treatment of the application of object-oriented technology into aviation software; however, the fundamental issues remain, with the constraints and additional hoops necessary to incorporate OOT still present. What is changing is the context. More and more manufacturers are biting the bullet and shifting to object-oriented technologies, establishing a clearer path to certifiability. Possibly even more significant, academia is increasingly deemphasizing structured methodologies, with structured design given only a brief nod as an introduction to the industry-favored object-oriented methodologies. Hiring and retaining engineering staff who will be satisfied working on the older technology will become increasingly challenging.

The question remains: should you make the jump to object-oriented technologies? Many factors will contribute to the calculus necessary to make the ultimate decision, including the cost of migration from your existing, instantiated platforms—not forgetting the associated retooling and retraining costs. The maturity of your staff will be a significant variable, aligning growth and maintenance considerations as engineers retire or leave, and must be replaced in a cost-effective manner. Cash flow and schedule constraints must also enter into the equation, with short-term migration impacts weighted against long-term stability and gain. Furthermore, a complete understanding of the guidance, its ramifications, and possible solutions will be critical in understanding the overall tradeoff: the true cost versus benefit of adopting OOT for your certifiable product development.

For many, the migration to OOT is inevitable, a necessary step to keep pace with the competition and maintain a solid foundation for future staffing. For the business manager, the timing of the migration—based on a clear view of the costs and benefits—will be the critical decision.
4. RTCA DO-333, Formal Methods Supplement to DO-178C and DO-278A

DO-333 defines formal methods as mathematically based techniques for the specification, development, and verification of computer systems. In other words, formal methods define a defensible formal process to model, describe, or verify some aspect of certifiable software. DO-333 suggests that formal methods can provide value to the certifiable-software community in many areas. Formal methods can allow precise and systematic communication among engineers—as well as providing verification confidence in freedom from exceptions and deadlocks, bounds on stack size during execution, and so forth.

Formal methods are described as two types of activities: modeling and analysis, with the model representing a given set of software aspects, which can then be used for analysis, simulation and/or code generation. Although the establishment of a formal model has intrinsic value in providing a systematic treatment of some aspect of software development, e.g., requirements development, the analysis of a formal model can provide significant value to the certifiable-software development process.

Formal methods were briefly introduced in DO-178B, but, in general, haven’t been considered sufficiently mature to provide widespread benefit. In the 20 years since the introduction of DO-178B, tools and techniques have advanced sufficiently to establish formal methods as a readily available tool in an aviation software development toolkit. Note that, with respect to analysis, the new guidance really represents a formal notification of the tightening of the certifiability guidance. For example, worse-case stack analysis has always been a necessary consideration of a certifiable-software development; but greater leeway in its justification, planning, and documentation has been enjoyed by the certifiable software engineering community. With the release of DO-178C, the FAA will demand more of the certifiable software developers with respect to formal and detailed treatment of the analyses presented as part of the certifiable-software verification efforts.

Therefore, the new guidance on formal methods must be approached with caution. Acceptable analytical verification performed in the past may no longer meet FAA muster. Formal methods—consisting of formal logic, discrete mathematics, and computer readable language—may be required. Simple documentation of experienced engineering judgment may no longer be considered sufficient. In summary, the new guidance on formal methods will impact your established, business-as-usual certification practices, as well as establish formal expectations for migrating to new, advanced model-based techniques and tools. Understanding the associated impacts with both gap analyses and detailed projections will be necessary to both shore up existing practices and support an effective migration to more competitive model-based certifiable software processes.
CONCLUSION
Without question, the new guidance adopted by the FAA will impact established certifiable-software development processes. Manufacturers will need to modify their existing practices to accommodate the revised guidance. Although the benefits from the adoption of new technologies and tools will be significant; expectations, especially in the short term, must be tempered by a clear vision of the challenges associated with migration and early adoption. Manufacturers must carefully consider both short- and long-term costs and benefits. A comprehensive and detailed gap analysis—together with a well-considered certifiable-development process roadmap—will be necessary for a certifiable product manufacturer to prepare for an effective treatment of the revised guidance. Avoiding costly rework, while effectively leveraging new technologies, is the key to remaining competitive. The time to plan is now; the costs of delay can be dramatic.

ABOUT FOLIAGE
Foliage, Inc., a product development company, partners with companies to address the business and technical challenges inherent in developing complex software-intensive systems. Providing a full complement of engineering and consulting services aligned to business needs and applied over the entire product lifecycle, Foliage enables companies to accelerate development and drive more predictability and productivity into their businesses.

Foliage leverages over 20 years of experience partnering with leading companies in the Medical and Life Sciences, Aerospace and Defense, and Industrial Equipment industries. Working with Foliage, our clients gain the critical insights necessary to develop products with a difference, and to deliver the products their customers want, when they want them.

Foliage is based in Burlington, Massachusetts with additional offices in California, the Netherlands and India. For more information, visit foliage.com.

168 Middlesex Turnpike, Burlington, MA 01803  781.993.5500