# You Keep Using That Word

Darren Cofer, Rockwell Collins Advanced Technology Center

Formal methods tools have been shown to be effective at finding defects in and verifying the correctness of safety-critical systems such as avionics systems. The recent release of DO-178C and the accompanying Formal Methods Supplement DO-333 will make it easier for developers of software for commercial aircraft to obtain certification credit for the use of formal methods. However, most developers of avionics systems are unfamiliar with formal methods, and most developers of formal methods tools are unfamiliar with certification requirements and processes. This article provides a brief overview of the certification process for commercial aircraft, as well as some of the issues related to the use of formal methods tools in this context.

## 1. INTRODUCTION

Certification. Verification. Qualification.

These are words that may appear in computer science and software publications. For example, there is active research related to certified compilers [Leroy 2006] and certifying model checkers [Dräger et al. 2010]. In these instances, the word *certification* is used in connection with the production of a proof certificate which may serve as evidence corroborating a specific analysis result or showing the correctness of a transformation. In other cases, certification may actually refer to a legal or regulatory process related to product acceptance or licensing by the government. Similarly, in some contexts *verification* is used to mean a formal proof of correctness, while in other contexts verification implies a manual code review or requirements-based testing. And *qualification* sounds like a straightfoward concept, but has a specific technical meaning in certain contexts.

So these words can have different meanings or implications in different contexts, which is fine, until these contexts overlap. In fact, this is exactly what is happening as we work toward greater adoption of formal methods tools in the development of safety-critical embedded systems. We may soon be using a certifying model checker to satisfy certification objectives for a flight control system, so we need to be careful about our definitions. Otherwise, we run the risk of looking like Vizzini in *The Princess Bride* [IMDB 1987], who repeatedly exclaims "Inconceivable!" unitl Inigo Montoya finally replies "You keep using that word. I do not think it means what you think it means."


Fig. 1. Certification! I do not think it means what you think it means.

There are a number of issues to be addressed before formal verification tools can be fully integrated into the design process for safety-critical systems. For example, most developers of avionics systems are unfamiliar with which formal methods tools are most appropriate for different problem domains. Different levels of expertise are necessary to use these tools effectively and correctly. Evidence must be provided of a formal method's soundness, a concept that is not well understood by most practicing engineers. Similarly, most developers of formal methods tools are unfamiliar with certification requirements and processes. DO-178C [RTCA 2011a] requires that a tool used to meet its objectives must be qualified in accordance with the tool qualification document DO-330 [RTCA 2011b]. The qualification of formal verification tools will likely pose unique challenges.

This article provides an overview of the concepts of certification, verification, and qualification, and how they relate to the use of formal methods tools. As a practical matter, we will focus on the civil aviation domain since there are published standards addressing the use of formal methods in the certification process. Similar notions of certification, software verification, and tool qualification are also found in the railway, nuclear, and medical device domains.

Much of the certification material that follows is adapted from [Bhattacharyya et al. 2015]. Additional information on formal methods and certification in commercial aircraft can be found on our research group's web site, Loonwerks.com.

## 2. CERTIFICATION

Certification is defined in DO-178C as legal recognition by the relevant certification authority that a product, service, organization, or person complies with its requirements. In the context of commercial aircraft, the relevant certification authority is the FAA in the U.S. or EASA in Europe. The requirements referred to are the government regulations regarding the airworthiness of aircraft operating in the National Airspace System (NAS). In practice, certification consists primarily of convincing representatives of a government agency that all required steps have been taken to ensure the safety, reliability, and integrity of the aircraft.

*Type certification* refers to approval of the aircraft design. Each aircraft manufactured is also individually certified to comply with its certified type design. Note that software itself is not certified in isolation, but only as part of an aircraft.

Certification differs from verification in that it focuses on evidence provided to a third party to demonstrate that the required activities were performed completely and correctly, rather on performance of the activities themselves. Also note that certification connects a product or design to legal requirements for its safety. Therefore, it is possible for a design to be safe but not certifiable. For example, the certification authority may for some reason not be convinced of the adequacy of the evidence provided.

### 2.1. Airworthiness Requirements

In the U.S., the legal requirements for aircraft operating in the NAS are defined in the Code of Federal Regulations, Title 14 (14CFR), *Aeronautics and Space.* The purpose of certification is to ensure that these legal requirements have been met.

Airworthiness standards for transport class aircraft are specified in Part 25 and standards for smaller aircraft are specified in Part 23. Parts 27 and 29 apply to rotorcraft and Part 33 to engines. Part 25 covers topics including Flight, Structure, Design and Construction, Powerplant, Equipment, Operating Limitations, and Electrical Wiring. Some of the requirements are quite detailed. For example, Subpart B (Flight) provides formulas and a detailed procedure for computing reference stall speed. It also provides requirements for controllability, trim conditions, and stability. Subpart D (Design and Construction) includes requirements for Control Systems related to stabil-

ity augmentation, trim systems, and limit load static tests. Some requirements cover items that no longer apply to modern aircraft (cables and pulleys).

### 2.2. Certification Process

The stakeholders in the civil aviation domain (FAA, airframers, equipment manufacturers) have developed a collection of documents defining a certification process which has been accepted as the standard means to comply with federal regulations. The process includes system development, safety assessment, and design assurance. These documents and their relationships are shown in Figure 2.

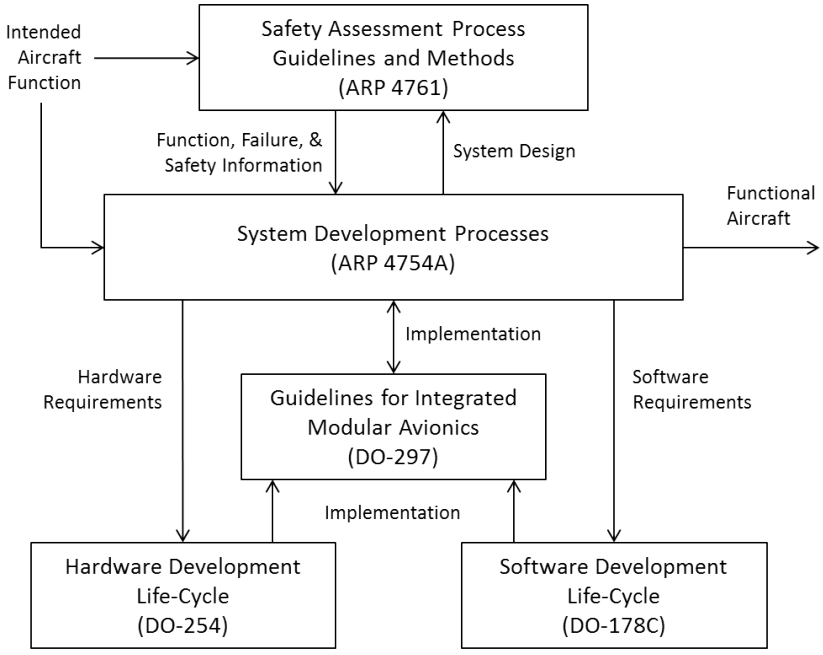


Fig. 2. Relationship among key documents in the certification process

The intended function, or requirements, for a new aircraft are the starting point for the process. These requirements are the basis for the aircraft system design that is produced in accordance with ARP4754A [SAE 2010], the guidelines for the system development process. The system design along with the aircraft requirements and its operating context are used to conduct a safety assessment in accordance with ARP4761 [SAE 1996].

The safety assessment determines, among other things, the criticality of system components as they contribute to the safety of the overall system. The system development process allocates functions and requirements to hardware and software components in the system, along with their assigned criticality from the safety assessment process. This information is used to develop the individual components and functions. The design assurance documents DO-178C (for software), DO-254 (for programmable hardware), and DO-297 (for integrated modular avionics) provide guidance for ensuring that these components satisfy the requirements that come from the system development process.

### 2.3. Safety Assessment

Safety assessment is performed in accordance with ARP4761, *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*. This document describes guidelines and methods for performing the safety assessment for certification of civil aircraft and is a means of showing compliance with the safety requirements of 14CFR. These requirements are hidden in Subpart F (Equipment) section 25.1309 with the unlikely title "Equipment, systems, and installations."

This section states that the equipment, systems, and installations required in an aircraft must be designed to ensure that they perform their intended functions under any foreseeable operating condition. The airplane systems and associated components, considered separately and in relation to other systems, must be designed so that:

— The occurrence of any failure condition which would prevent the continued safe flight and landing of the airplane is extremely improbable, and
— The occurrence of any other failure conditions which would reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions is improbable.

The section goes on to state that warning information must be provided to alert the crew to unsafe system operating conditions, and that systems, controls, and associated monitoring and warning means must be designed to minimize crew errors which could create additional hazards. Compliance must be shown by analysis or testing that considers possible modes of failure (including malfunctions and damage from external sources), the probability of multiple failures and undetected failures, the resulting effects on the airplane and occupants, and the crew warning cues, corrective action required, and the capability of detecting faults.

### 2.4. System Development

Aircraft system development is described in ARP4754A, *Guidelines for Development of Civil Aircraft and Systems*. This document discusses the development of aircraft systems, taking into account the overall aircraft operating environment and functions. This includes validation of requirements and verification of the design implementation for certification and product assurance. It provides practices for showing compliance with the regulations.

ARP4754A provides guidance for creating plans for the system development and eight *integral processes* which span all of the system development activities. The integral processes are safety assessment, assurance level assignment, requirements capture, requirements validation, implementation verification, configuration management, process assurance, and certification and regulatory authority coordination. The system development process allocates functionality and defines requirements for components, both hardware and software. It invokes the safety assessment process and ensures that the system design satisfies safety requirements for the aircraft. It also guides developers in allocating system requirements to hardware and software components and in determining the criticality level for those components.

## 3. VERIFICATION

The software assurance process makes sure that components are developed to meet their requirements *without any unintended functionality.* This means that the process will include activities specifically designed to provide evidence that the software does only what its requirements specify and nothing else.

For software in commercial aircraft, the relevant guidance is found in DO-178C, *Software Considerations in Airborne Systems and Equipment Certification.* Certification authorities in North American and Europe have agreed that an applicant (aircraft manufacturer) can use this guidance as a means of compliance with the regulations governing aircraft certification.

The original version of the document, DO-178, was approved in 1982 and consisted largely of a description of best practices for software development. It was revised in 1985 as DO-178A, adding definitions of three levels of software criticality, with development and verification processes described in more detail. DO-178B, approved in 1992, defined five levels of software criticality (A – E, with level A being the most critical) with specific objectives, activities, and evidence required for each level. The processes and objectives in the document assume a traditional development process with test-based verification.

In 2005, the publishers of DO-178 initiated work on a revision to be known as DO-178C. A committee was chartered to draft the new document, with the objectives of minimizing changes to the core document, yet updating it to accommodate approximately 15 years of progress in software engineering. Guidance specific to new software technologies was to be contained in supplements which could add, modify, or replace objectives in the core document. New supplements were developed in the areas of object-oriented design, model-based development, and formal methods, as well as an additional document containing new guidance on tool qualification. DO-178C and its associated documents were published in 2011 and accepted by the FAA as a means of compliance in 2013.

### 3.1. Software Development

DO-178C does not prescribe a specific development process, but instead identifies important activities and design considerations throughout a development process and defines objectives for each of these activities. It assumes a traditional development process that can be decomposed as follows:

— Software Requirements Process. Develops High Level Requirements (HLR) from the output of the system design process.
— Software Design Process. Develops Low Level Requirements (LLR) and Software Architecture from the HLR.
— Software Coding Process. Develops source code from the Software Architecture and the LLR.
— Software Integration Process. Combines executable object code modules with the target hardware for hardware/software integration.

Each of these processes produces or updates a collection of artifacts, culminating in an integrated executable (see Figure 3).

### 3.2. Software Verification

The results of these processes are verified through the verification process. The verification process consists of review, analysis, and test activities that must provide evidence of the correctness of the development activities.

In general, verification has two complementary objectives. One objective is to demonstrate that the software satisfies its requirements. The second objective is to demonstrate with a high degree of confidence that errors which could lead to unacceptable failure conditions, as determined by the system safety assessment process, have been removed.

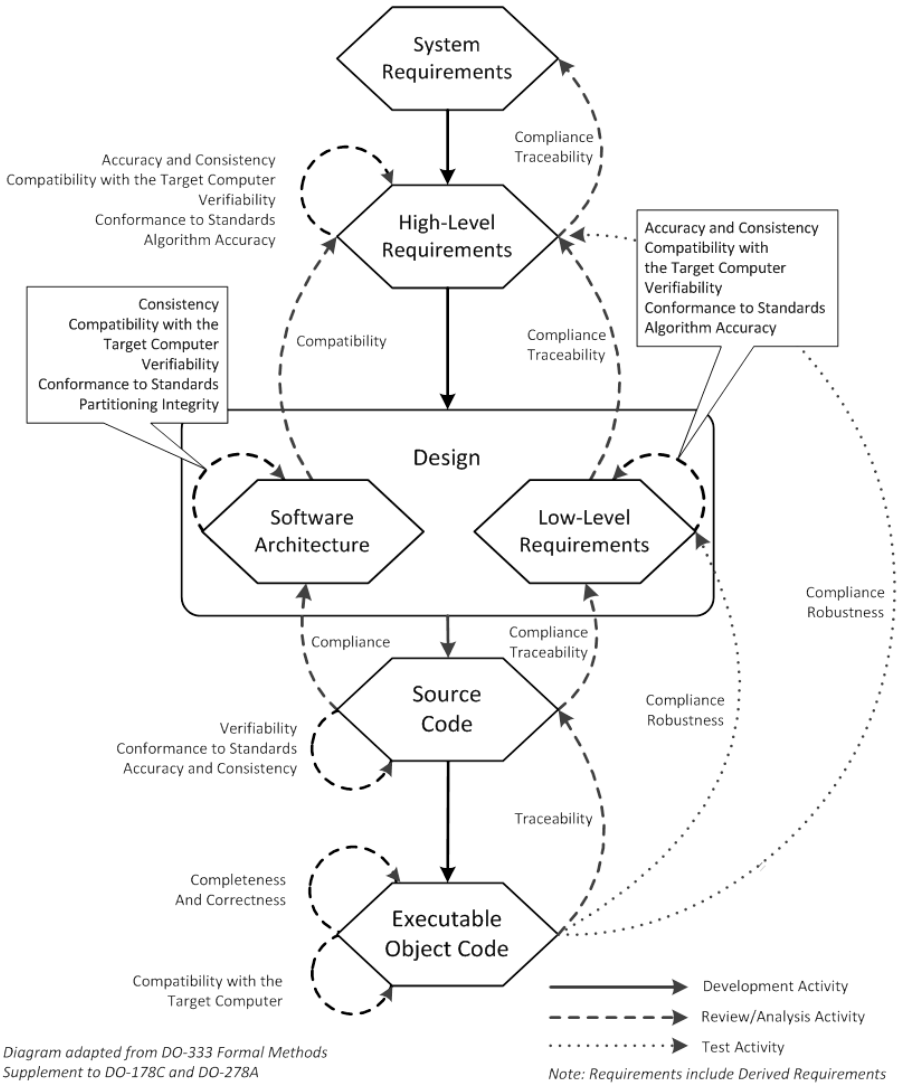


Fig. 3. DO-178C certification activities required for Level A code.

One of the foundational principles of DO-178C is requirements-based testing. This means that the verification activities are centered around explicit demonstration that each requirement has been met.

A second principle is complete coverage, both of the requirements and of the code that implements them. This means that every requirement and every line of code will be examined in the verification process. Furthermore, several metrics are defined which specify the degree of structural coverage that must be obtained in the verification process, depending on the criticality of the software being verified.

A third principle is traceability among all of the artifacts produced in the development process. This means that:

— Every requirement must have one or more associated test cases. All testing must trace to a specific requirement.
— Every requirement must be traceable to code that implements it. Every line of code must be traceable to a requirement.
— Every line of code (and, in some cases, every branch and condition in the code) must be exercised by a test case.

Together, these objectives provide evidence that all requirements are correctly implemented and that no unintended function has been implemented.

Of particular interest is DO-333, the Formal Methods Supplement to DO-178C [RTCA 2011c]. DO-333 extends the guidance provided in DO-178C and describes how formal methods may be used to satisfy its certification objectives. Several case studies showing examples of how to use different formal verification tools to satisfy various certification objectives are available in [Cofer and Miller 2014]. DO-333 generally allows the testing described above to be replaced by a comparable formal analysis. However, even when formal methods are used some on-target testing is still required.

One constraint imposed by ARP4754A (and DO-178C) is that requirements must be *verifiable*, which in the past has meant *testable*. This meant that in practice there could be no negative requirements such those related to safety (e.g., "The system can never enter an unsafe state.") However, with the advent of DO-333, such requirements can now be addressed analytically and may be very useful in demonstrating the safety of a complex avionics system.

## 4. QUALIFICATION

Tool qualification is the process by which certification credit may be claimed for the use of a software tool. Qualification is required whenever a certification process is eliminated, reduced, or automated by a software tool without its output being verified. The purpose of tool qualification is to ensure that the tool provides confidence at least equivalent to that of the process it eliminates, reduces, or automates. Tool qualification is, therefore, a significant aspect of any certification effort.

Software tools are used in development processes to automate life cycle activities that are complex and error-prone if performed by humans. The use of such tools should, in principle, be encouraged from a certification perspective to provide confidence in the correctness of the software product. Therefore, we should avoid unnecessary barriers to tool qualification which may inadvertently reduce the use of tools that would otherwise enhance software quality and confidence.

Formal methods tools have matured to the point where they are capable of analyzing software systems of practical size, and their effectiveness in finding errors has been demonstrated repeatedly [Woodcock et al. 2009], [Miller et al. 2010]. Commercial tools used in aerospace and other safety-critical domains are beginning to include formal verification capabilities. For example, MATLAB now markets both Simulink Design Verifier, a model checker, and Polyspace, an abstract interpretation tool. Esterel Technologies includes a model checker, Design Verifier, as part of their SCADE Suite toolset.

If formal verification is used to satisfy DO-178C objectives, DO-333 requires the applicant to provide evidence that the underlying method is *sound,* i.e., it will never prove a property to be true when it is actually false. In addition, if the formal verification is to be implemented in a software tool, the tool must be qualified in accordance with DO-330. While clearly related, the concepts of tool qualification and soundness of the underlying method were intentionally kept separate by the standards' authors.

DO-330 defines five tool qualification levels (TQL) ranging from TQL-1 for software development tools that generate Level A source code to TQL-5 for software verification

tools. The TQL is determined both by the criticality of the software the tool is being used on and the impact of the tool on the software development process. A strong distinction is made between a development that could potentially insert an error into the embedded software, and a verification tool that could fail to detect an error. A trusted compiler or code generator would be classified as TQL-1 through 4, depending on the criticality of the code it is used to generate. Formal verification tools are classified as TQL-4 or TQL-5.

Despite the additional guidance provided in DO-178C, DO-330, and DO-333, there are still many questions to be addressed. For one thing, most practicing engineers are unaware of how to apply different categories of formal verification tools. Even within a particular category, there are a wide variety of tools, often based on fundamentally different approaches, each with its own strengths and weaknesses. For example, an explicit state model checker operates in a fundamentally different way from an SMT (Satisfiability Modulo Theories) based model checker.

Typically, a tool will be shown to meet its requirements through testing, analysis, and reviews, just as for airborne or ground-based software developed in accordance with DO-178C. However, formal verification tools differ from many tools in that they are typically "exhaustive" and cover all combinations of inputs and state. Development of the tool operational requirements and test cases for such tools may pose unique challenges.

Using a formal verification tool to meet DO-178C objectives may require more than just qualification of the tool itself. For example, it is frequently necessary to translate a software model (e.g., a Simulink model) to the input language of the verification tool. In such cases, consideration must be given as to why that translation is to be trusted. Is the translation included as part of the tool operational requirements and verified as part of the tool qualification, or are the translator and the verification tool regarded as two separate entities? Outputs of the formal verification tool often need to be translated back to a representation the system developers can understand and similar questions apply to why this translation can be trusted. Potential user errors must be considered. Are the users allowed to introduce assumptions about the environment of the unit being checked, and if so, are these clearly identified and validated? Are there tool configuration settings or modes of operation that can cause it to generate unsound results? All ways in which use of the tool might provide false confidence need to be identified and accounted for.

At the same time, it is also important to not make the cost of qualification of formal methods tools so great as to discourage their use. While it is tempting to hold formal verification tools to a higher standard than other software tools, making their qualification unnecessarily expensive could do more harm than good.

## 5. CONCLUSION

Formal methods tools have the potential to ensure the quality of safety-critical systems by providing comprehensive evaluation of the behavior of complex embedded software. They have also been shown to reduce costs through the early detection and elimination of design errors. Improved communication between formal methods researchers, software developers, and certification authorities will be an important enabler in the continued adoption of formal methods in industries such as aerospace that have strong certification requirements. With a shared understanding of this context, our expectations for the use of formal methods to satisfy certification objectives are changing from "Inconceivable!" to "Of course!"

## ACKNOWLEDGMENTS

Thanks to Deb Turcio at DebbieDrawsFunny.com for permission to use the drawing in Figure 1.

## REFERENCES

Siddhartha Bhattacharyya, Darren Cofer, David J. Musliner, Joseph Mueller, and Eric Engstrom. 2015. *Certification Considerations for Adaptive Systems*. Technical Report NASA/CR2015-218702. NASA Contractor Report.

Darren Cofer and Steven P. Miller. 2014. *Formal Methods Case Studies for DO-333*. Technical Report NASA/CR-2014-218244. NASA.

Klaus Dräger, Andrey Kupriyanov, Bernd Finkbeiner, and Heike Wehrheim. 2010. SLAB: A Certifying Model Checker for Infinite-State Concurrent Systems. In *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*. 271–274.

IMDB. 1987. The Princess Bride. (1987). http://www.imdb.com/title/tt0093779/

Xavier Leroy. 2006. Formal Certification of a Compiler Back-end or: Programming a Compiler with a Proof Assistant. In *Proceedings of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2006, Charleston SC, USA, January 11-13, 2006*. 42–54.

Steven P. Miller, Michael W. Whalen, and Darren D. Cofer. 2010. Software model checking takes off. *Commun. ACM* 53, 2 (2010), 58–64.

RTCA. 2011a. DO-178C, *Software Considerations in Airborne Systems and Equipment Certification*. (2011).

RTCA. 2011b. DO-330, *Software Tool Qualification Considerations*. (2011).

RTCA. 2011c. DO-333, *Formal Methods Supplement to DO-178C and DO-278A*. (2011).

SAE. 1996. ARP4761, *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*. (1996).

SAE. 2010. ARP4754A, *Guidelines For Development Of Civil Aircraft and Systems on Civil Airborne Systems and Equipment*. (2010).

Jim Woodcock, Peter Gorm Larsen, Juan Bicarregui, and John S. Fitzgerald. 2009. Formal Methods: Practice and Experience. *ACM Comput. Surv.* 41, 4 (2009).