

RTCA DO-178B/EUROCAE ED-12B

Thomas K. Ferrell

Ferrell and Associates Consulting

Uma D. Ferrell

Ferrell and Associates Consulting

27.1 Introduction

Comparison with Other Software Standards • Document Overview • Software as Part of the System

27.2 Software Life-Cycle Processes

Software Planning Process • Software Development Process

27.3 Integral Process

Software Verification • Software Configuration Management • Software Quality Assurance • Certification Liaison Process

27.4 Additional Considerations

Previously Developed Software • Tool Qualification

27.5 Additional Guidance

27.6 Synopsis

References

Further Information

27.1 Introduction

This chapter provides a summary of the document RTCA DO-178B, *Software Considerations in Airborne Systems and Equipment Certification*,¹ with commentary on the most common mistakes made in understanding and applying DO-178B. The joint committee of RTCA Special Committee 167 and EUROCAE* Working Group 12 prepared RTCA DO-178B** (also known as EUROCAE ED-12B), and it was subsequently published by RTCA and by EUROCAE in December 1992. DO-178B provides guidance for the production of software for airborne systems and equipment such that there is a level of confidence in the correct functioning of that software in compliance with airworthiness requirements. DO-178B represents industry consensus opinion on the best way to ensure safe software. It should also be noted that although DO-178B does not discuss specific development methodologies or management activities, there is clear evidence that by following rigorous processes, cost and schedule benefits may be realized. The verification activities specified in DO-178B are particularly effective in identifying software problems early in the development process.

*European Organization for Civil Aviation Equipment.

**DO-178B and ED-12B are copyrighted documents of RTCA and EUROCAE, respectively. In this chapter, DO-178B shall be used to refer to both the English version and the European equivalent. This convention was adopted solely as a means for brevity.

27.1.1 Comparison with Other Software Standards

DO-178B is a mature document, having evolved over the last 20 years through two previous revisions, DO-178 and DO-178A. It is a consensus document that represents the collective wisdom of both the industry practitioners and the certification authorities. DO-178B is self-contained, meaning that no other software standards are referenced except for those that the applicant produces to meet DO-178B objectives. Comparisons have been made between DO-178B and other software standards such as MIL-STD-498, MIL-STD-2167A, IEEE/EIA-12207, IEC 61508, and U.K. Defense Standard 0-55. All of these standards deal with certain aspects of software development covered by DO-178B. None of them has been found to provide complete coverage of DO-178B objectives. In addition, these other standards lack objective criteria and links to safety analyses at the system level. However, organizations with experience applying these other standards often have an easier path to adopting DO-178B.

Advisory Circular AC 20-115B specifies DO-178B as an acceptable means, but not the only means, for receiving regulatory approval for software in systems or equipment being certified under a Technical Standard Order (TSO) Authorization, Type Certificate (TC), or Supplemental Type Certificate (STC). Most applicants use DO-178B to avoid the work involved in showing that other means are equivalent to DO-178B. Even though DO-178B was written as a guideline, it has become the standard practice within the industry. DO-178B is officially recognized as a *de facto* international standard by the International Organization for Standardization (ISO).

27.1.2 Document Overview

DO-178B consists of 12 sections, 2 annexes, and 3 appendices as shown in [Figure 27.1](#).

Section 2 and Section 10 are designed to illustrate how the processes and products discussed in DO-178B relate to, take direction from, and provide feedback to the overall certification process. Integral Processes detailed in Sections 6, 7, 8, and 9, support the software life cycle processes noted in Sections 3, 4, and 5.

Section 11 provides details on the life cycle data and Section 12 gives guidance to any additional considerations. Annex A, discussed in more detail below, provides a leveling of objectives. Annex B provides the document's glossary. The glossary deserves careful consideration since much effort and care was given to precise definition of the terms. Appendices A, B, C, and D provide additional material

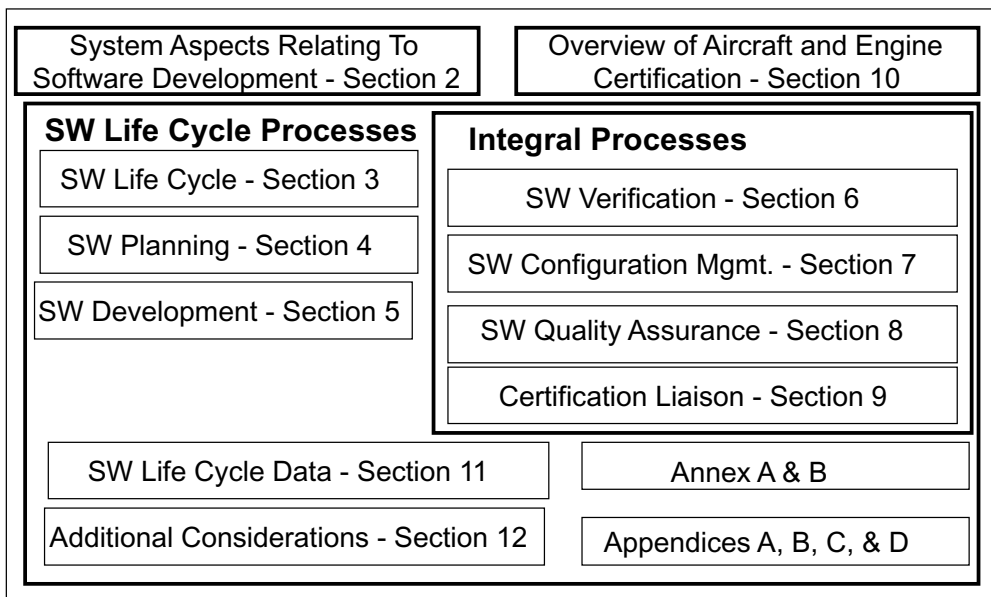


FIGURE 27.1 Document structure.

Objective		Applicability by SW Level				Output		Control Category by SW Level				
	Description	Ref.	A	B	C	D	Description	Ref.	A	B	C	D
1	Low-level Requirements comply with High-level Requirements.	6.3.2 a	●	●	○		Software Verification Results	11.14	②	②	②	②

FIGURE 27.2 Example objective from Annex A.

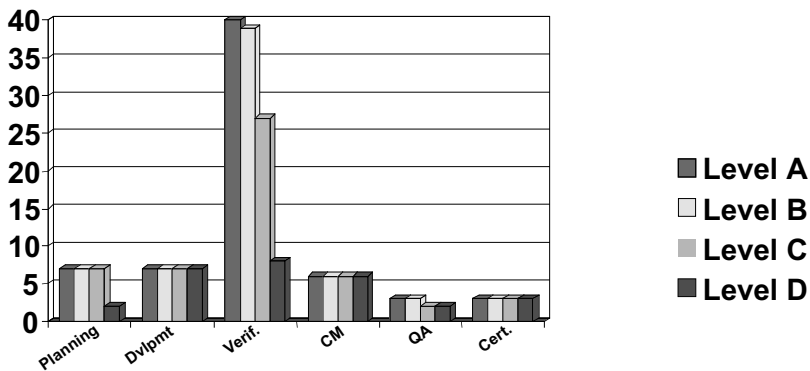


FIGURE 27.3 Objectives over the software development life cycle.

including a brief history of the document, the index, a list of contributors, and a process improvement form, respectively. It is important to note that with the exception of the appendices and some examples embedded within the text, the main sections and the annexes are considered normative, i.e., required to apply DO-178B.

The 12 sections of DO-178B describe processes and activities for the most stringent level of software.* Annex A provides a level by level tabulation of the objectives for lower levels of software.** This leveling is illustrated in Figure 27.2 extracted from Annex A Table A-4, Verification of Outputs of Software Design Process.

In addition to the leveling of objectives, Annex A tables serve as an index into the supporting text by way of reference, illustrate where independence is required in achieving the objective, which data items should include the objective evidence, and how that evidence must be controlled. More will be said about the contents of the various Annex A tables in the corresponding process section of this text. If an applicant adopts DO-178B for certification purposes, Annex A may be used as a checklist to achieve these objectives. The FAA's position is that if an applicant provides evidence to satisfy the objectives, then the software is DO-178B compliant. Accordingly, the FAA's checklists for performing audits of DO-178B developments are based on Annex A tables.

Before discussing the individual sections, it is useful to look at a breakout of objectives as contained in Annex A. While DO-178B contains objectives for the entire software development life cycle, there is a clear focus on verification as illustrated by Figure 27.3. Although at first glance it appears that there is only one objective difference between levels A and B, additional separation between the two is accomplished through

*Levels are described in Section 27.1.3, "Software as part of system."

**Software that is determined to be at level E is outside the scope of DO-178B.

the relaxation of independence requirements. Independence is achieved by having the verification or quality assurance of an activity performed by a person other than the one who initially conducted the activity. Tools may also be used to achieve independence.

27.1.3 Software as Part of the System

Application of DO-178B fits into a larger system of established or developing industry practices for systems development and hardware. The system level standard is SAE ARP4754, *Certification Considerations for Highly-Integrated or Complex Aircraft Systems*.² The relationship between system, software, and hardware processes is illustrated in Figure 27.4.

The interfaces to the system development process were not well defined at the time DO-178B was written. This gap was filled when ARP4754 was published in 1996. DO-178B specifies the information flow between system processes and software processes. The focus of the information flow from the system process to the software process is to keep track of requirements allocated to software, particularly those requirements that contribute to the system safety. The focus of information flow from the software process to the system process is to ensure that changes in the software requirements, including the introduction of derived requirements (those not directly traceable to a parent requirement), do not adversely affect system safety.

The idea of system safety, although outside the scope of DO-178B, is crucial to understanding how to apply DO-178B. The regulatory materials governing the certification of airborne systems and equipment define five levels of failure conditions. The most severe of these is *catastrophic*, meaning failures that result in the loss of ability to continue safe flight and landing. The least severe is *no effect*, where the failure results in no loss of operational capabilities and no increase in crew workload. The intervening levels define various levels of loss of functionality resulting in corresponding levels of workload and potential for loss of life. These five levels map directly to the five levels of software defined in DO-178B. This mapping is shown in Figure 27.5.

It is important to note that software is never certified as a standalone entity. A parallel exists for the hardware development process and flow of information between hardware processes and system process. Design trade-offs between software processes and hardware processes are also taken into consideration

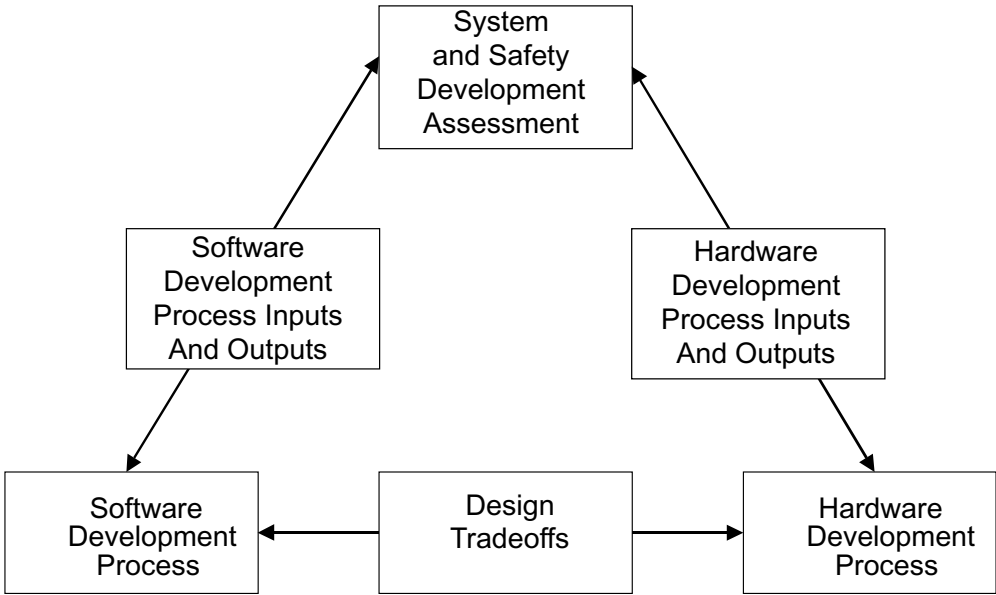


FIGURE 27.4 Relationship between system development process and the software development process.

Failure Condition	DO-178B Software Level
Catastrophic	A
Hazardous	B
Major	C
Minor	D
No effect	E

FIGURE 27.5 Software levels.

at the system level. Software levels may be lowered by using protective software or hardware mechanisms elsewhere in the system. Such architectural methods include partitioning, use of hardware or software monitors, and architectures with built-in redundancy.

27.2 Software Life-Cycle Processes

The definition of how data are exchanged between the software and systems development processes is part of the software life-cycle processes discussed in DO-178B. Life-cycle processes include the planning process, the software development processes (requirements, design, code, and integration), and the integral processes (verification, configuration management, software quality assurance, and certification liaison). DO-178B defines objectives for each of these processes as well as outlining a set of activities for meeting the objectives.

DO-178B discusses the software life-cycle processes and transition criteria between life-cycle processes in a generic sense without specifying any particular life-cycle model. Transition criteria are defined as “the minimum conditions, as defined by the software planning process, to be satisfied to enter a process.” Transition criteria may be thought of as the interface points between all of the processes discussed in DO-178B. Transition criteria are used to determine if a process may be entered or reentered. They are a mechanism for knowing when all of the tasks within a process are complete and may be used to allow processes to execute in parallel. Since different development models require different criteria to be satisfied for moving from one step to the next, specific transition criteria are not defined in DO-178B. However, it is possible to describe a set of characteristics that all well-defined transition criteria should meet. For transition criteria to successfully assist in entry from one life-cycle process to another, they should be quantifiable, flexible, well documented, and present for every process. It is also crucial that the process owners agree upon the transition criteria between their various processes.

27.2.1 Software Planning Process

DO-178B defines five types of planning data* for a software development. They are

- Plan for Software Aspects of Certification (PSAC)
- Software Development Plan
- Software Verification Plan

*The authors of DO-178B took great pains to avoid the use of the term “document” when referring to objective evidence that needed to be produced to satisfy DO-178B objectives. This was done to allow for alternative data representations and packaging as agreed upon between the applicant and the regulatory authority. For example, the four software plans-outlining development, verification, QA, and CM may all be packaged in a single plan, just as the PSAC may be combined with the System Certification Plan.

- Software Configuration Management Plan
- Software Quality Assurance Plan

These plans should include consideration of methods, languages, standards, and tools to be used during the development. A review of the planning process should have enough details to assure that the plans, proposed development environment, and development standards (requirements, design, and code) comply with DO-178B.

Although DO-178B does not discuss the certification liaison process until Section 9, the intent is that the certification liaison process should begin during the projects' planning phase. The applicant outlines the development process and identifies the data to be used for substantiating the means of compliance for the certification basis. It is especially important that the applicant outline specific features of software or architecture that may affect the certification process.

27.2.2 Software Development Process

Software development processes include requirements, design, coding, and integration. DO-178B allows for requirements to be developed that detail the system's functionality at various levels. DO-178B refers to these levels as high- and low-level requirements. System complexity and the design methodology applied to the system's development drive the requirements' decomposition process. The key to understanding DO-178B's approach to requirement's definition can be summed up as, "one person's requirements are another person's design." Exactly where and to what degree the requirements are defined is less important than ensuring that all requirements are accounted for in the resulting design and code, and that traceability is maintained to facilitate verification.

Some requirements may be derived from the design, architecture, or the implementation nuances of the software and hardware. It is recognized that such requirements will not have a traceability to the high-level requirements. However, these requirements must be verified and must also be considered for safety effects in the system safety assessment process.

DO-178B provides only a brief description of the design, coding, and integration processes since these tend to vary substantially between various development methodologies. The one exception to this is in the description to the outputs of each of the processes. The design process yields low-level requirements and software architecture. The coding process produces the source code, typically either in a high-order language or assembly code. The result of the integration effort is executable code resident on the target computer along with the various build files used to compile and link the executable. Each of these outputs is verified, assured, and configured as part of the integral processes.

27.3 Integral Processes

DO-178B defines four processes as integral, meaning that they overlay and extend throughout the software life cycle. These are the software verification process, software configuration management, software quality assurance, and certification liaison process.

27.3.1 Software Verification*

As noted earlier, verification objectives outnumber all others in DO-178B, accounting for over two thirds of the total. DO-178B defines verification as a combination of reviews, analyses, and testing. Verification is a technical assessment of the results of both the software development processes and the software verification process. There are specific verification objectives that address the requirements, design, coding, integration, as well as the verification process itself. Emphasis is placed at all stages to assure that there is traceability from high-level requirements to the final "as-built" configuration.

*Software Verification is a complex topic, which deserves in-depth treatment. The reader is directed to References 4, 5, and 6 for detailed discussion on verification approaches and explanation of terms.

Reviews provide qualitative assessment of a process or product. The most common types of reviews are requirements reviews, design reviews, and test procedure reviews. DO-178B does not prescribe how these reviews are to be conducted, or what means are to be employed for effective reviews. Best practices in software engineering process states that for reviews to be effective and consistent, checklists should be developed and used for each type of review. Checklists provide:

- Objective evidence of the review activity
- A focused review of those areas most prone to error
- A mechanism for applying “lessons learned”
- A practical traceable means for ensuring that corrective action is taken for unsatisfactory items

Review checklists can be common across projects, but they should themselves be reviewed for appropriateness and content for a particular project.

Analyses provide repeatable evidence of correctness and are often algorithmic or procedural in nature. Common types of analyses used include timing, stack, data flow, and control flow analyses. Race conditions and memory leakage should be checked as part of the timing and stack analysis. Data and control coupling analysis should include, a minimum, basic checks for set/use and may extend to a full model of the system’s behavior. Many types of analyses may be performed using third-party tools. If tools are used for this purpose, DO-178B rules for tool qualification must be followed.

The third means of verification, testing, is performed to demonstrate that

- The software product performs its intended function
- The software does not demonstrate any unintended actions

The key to accomplishing testing correctly to meet DO-178B objectives in a cost-effective manner is to maintain a constant focus on requirements. This requirements-based test approach represents one of the most fundamental shifts from earlier versions of the document. As test cases are designed and conducted, requirements coverage analysis is performed to assess that all requirements are tested. A structural coverage analysis is performed to determine the extent to which the requirements-based test exercised the code. In this manner, structural coverage is used as a means of assessing overall test completion. The possible reasons for lack of structural coverage are shortcomings in requirements-based test cases or procedures, inadequacies in software requirements, compiler generated code, unreachable, or inactive code.

As part of the test generation process, tests should be written for both normal range and abnormal inputs (robustness). Tests should also be conducted using the target environment whenever possible.

Structural coverage and how much testing is required for compliance at the various levels are misunderstood topics. Level D software verification requires test coverage of high-level requirements only. No structural coverage is required.

Low-level requirements testing is required at level C. In addition, testing of the software structure to show proper data and control coupling is introduced. This coverage involves coverage of dependencies of one software component on other software component via data and control. Decision coverage is required for level B, while level A code requires Modified Condition Decision Coverage (MCDC).

For level A, structural coverage analysis may be performed on source code only to the extent that the source code can be shown to map directly to object code. The reason for this rule is that some compilers may introduce code or structure that is different from source code.

MCDC coverage criteria were introduced to retain the benefits of multiple-condition coverage while containing the exponential growth in the required number of test cases required. MCDC requires that each condition must be shown to independently affect the outcome of the decision and that the outcome of a decision changes when one condition is changed at a time. Many tools are available to determine the minimum test case set needed for DO-178B compliance. There is usually more than one set of test cases that satisfy MCDC coverage.³ There is no firm policy on which set should be used for compliance. It is best to get an agreement with the certification authorities concerning the algorithms and tools used to determine compliance criteria.

27.3.2 Software Configuration Management

Verification of the various outputs discussed in DO-178B are only credible when there is clear definition of what has been verified. This definition or configuration is the intent of the DO-178B objectives for configuration management. The six objectives in this area are unique, in that they must be met for all software levels. This includes identification of what is to be configured, how baselines and traceability are established, how problem reports are dealt with, how the software is archived and loaded, and how the development environment is controlled.

While configuration management is a fairly well-understood concept within the software engineering community (as well as the aviation industry as a whole), DO-178B does introduce some unique terminology that has proven to be problematic. The concept of control categories is often misunderstood in a way that overall development costs are increased, sometimes dramatically. DO-178B defines two control categories (CC1 and CC2) for data items produced throughout the development.

The authors of DO-178B intended the two levels as a way of controlling the overhead costs of creating and maintaining the various data items. Items controlled as CC2 have less requirements to meet in the areas of problem reporting, baselining, change control, and storage. The easiest way to understand this is to provide an example. Problem reports are treated as a CC2 item. If problem reports were a CC1 item and a problem was found with one of the entries on the problem report itself, a second problem report would need to be written to correct the first one.

A second nuance of control categories is that the user of DO-178B may define what CC1 and CC2 are within their own CM system as long as the DO-178B objectives are met. One example of how this might be beneficial is in defining different retention periods for the two levels of data. Given the long life of airborne systems, these costs can be quite sizeable. Another consideration for archival systems selected for data retention is technology obsolescence of the archival medium as well as means of retrieval.

27.3.3 Software Quality Assurance

Software quality assurance (SQA) objectives provide oversight of the entire DO-178B process and require independence at all levels. It is recognized that it is prudent to have an independent assessment of quality. SQA is active from the beginning of the development process. SQA assures that any deviations during the development process from plans and standards are detected, recorded, evaluated, tracked, and resolved. For levels A and B, SQA is required to assure transition criteria are adhered to throughout the development process.

SQA works with the CM process to assure that proper controls are in place and applied to life cycle data. This last task culminates in the conduct of a software conformity review. SQA is responsible for assuring that the as-delivered products matches the as-built and as-verified product. The common term used for this conformity review in commercial aviation industry is “First Article Inspection.”

27.3.4 Certification Liaison Process

As stated earlier, the certification liaison process is designed to streamline the certification process by ensuring that issues are identified early in the process. While DO-178B outlines twenty distinct data items to be produced in a compliant process, three of these are specific to this process and must be provided to the certifying authority. They are

- Plan for Software Aspects of Certification (PSAC)
- Software Configuration Index
- Software Accomplishment Summary

Other data items may be requested by the certification authority, if deemed necessary. As mentioned earlier, applicants are encouraged to start a dialogue with certification authorities as early in the process as possible to reach a common understanding of a means of achieving compliance with DO-178B. This is especially important as new technology is applied to avionics and as new personnel enter the field.

Good planning up front, captured in the PSAC, should minimize surprises later in the development process, thus minimizing cost. Just as the PSAC states *what you intend to do*, the accomplishment summary captures *what you did*. It is used to gauge the overall completeness of the development process and to ensure that all objectives of DO-178B have been satisfied.

Finally, the configuration index serves as an overall accounting of the content of the final product as well as the environment needed to recreate it.

27.4 Additional Considerations

During the creation of DO-178B, it was recognized that new development methods and approaches existed for developing avionics. These included incorporation of previously developed software, use of tools to accomplish one or more of the objectives required by DO-178B, and application of alternate means in meeting an objective such as formal methods. In addition, there are a small class of unique issues such as field-loadable and user-modifiable software. Section 12 collects these items together under the umbrella title of Additional Considerations. Two areas, Previously Developed Software (PDS) and Tool Qualification, are common sources of misunderstanding in applying DO-178B.

27.4.1 Previously Developed Software

PDS is software that falls in one of the following categories:

- Commercial off-the-shelf software (e.g., shrink-wrap)
- Airborne software developed to other standards (e.g., MIL-STD-498)
- Airborne software that predates DO-178B (e.g., developed to the original DO-178 or DO-178A)
- Airborne software previously developed at a lower software level

The use of one or more of these types of software should be planned for and discussed in the PSAC. In every case, some form of gap analysis must be performed to determine where specific objectives of DO-178B have not been met. It is the applicant's responsibility to perform this gap analysis and propose to the regulatory authority a means for closing any gaps. Alternate sources of development data, service history, additional testing, reverse engineering, and wrappers* are all ways of ensuring the use of PDS is safe in the new application.

In all cases, usage of PDS must be considered in the safety assessment process and may require that the process be repeated if the decision to use a PDS component occurs after the approval of PSAC. A special instance of PDS usage occurs when software is used in a system to be installed on an aircraft other than the one for which it was originally designed. Although the function may be the same, interfaces with other aircraft systems may behave differently. As before, the system safety assessment process must be repeated to assure that the new installation operates and behaves as intended.

If service history is employed in making the argument that a PDS component is safe for use, the relevance and sufficiency of the service history must be assessed. Two tests must be satisfied for the service history approach to work. First, the application for which history exists must be shown to be similar to the intended new use of the PDS. Second, there should be data, typically problem reports, showing how the software has performed over the period for which credit is sought. The authors of DO-178B intended that any use of PDS be shown to meet the same objectives required of newly developed code.

Prior to identifying PDS as part of a new system, it is prudent to investigate and truly understand the costs of proving that the PDS satisfies the DO-178B objectives. Sometimes, it is easier and cheaper to develop the code again!

*Wrappers is a generic term used to refer to hardware or software components that isolate and filter inputs to and from the PDS for the purposes of protecting the system from erroneous PDS behavior.

27.4.2 Tool Qualification

DO-178B requires qualification of tools when the processes noted by DO-178B are eliminated, reduced, or automated by a tool without its output being verified according to DO-178B. If the output of a tool is demonstrated to be restricted to a particular part of the life cycle, the qualification can also be limited to that part of the life cycle. Only deterministic tools can be qualified.

Tools are classified as development tools and verification tools. Development tools produce output that becomes a part of the airborne system and thus can introduce errors. Rules for qualifying development tools are fashioned after the rules of assurance for generating code. Once the need for development tool qualification is established, a tool qualification plan must be written. The rigor of the plan is determined by the nature of the tool and the level of code upon which it is being used. A tool accomplishment summary is used to show compliance with the tool qualification plan. The tool is required to satisfy the objectives at the same level as the software it produces, unless the applicant can justify a reduction in the level to the certification authority.

Verification tools cannot introduce errors but may fail to detect them or mask their presence. Qualification criterion for verification tools is the demonstration of its requirements under normal operational conditions. Compliance is established by noting tool qualification within PSAC and Software Accomplishment Summary. A tool qualification plan and a tool accomplishment summary are not required for verification tools by DO-178B although an applicant may find them useful for documenting the qualification effort.

27.5 Additional Guidance

RTCA SC-190/EUROCAE WG-52 was formed in 1997 to address issues that were raised by the industry and certification authorities in the course of applying DO-178B since its release in 1992. The membership in this committee includes over 200 industry and regulatory representatives from the U.S. and Europe. The outputs of the SC-190 consensus process are available to industry from the RTCA or EUROCAE in the form of errata, frequently asked questions, and discussion papers. These outputs have been collated and published in DO-248/ED-94.⁷

27.6 Synopsis

DO-178B provides objectives for software life-cycle processes, activities to achieve these objectives, and outlines objective evidence for demonstrating that these objectives were accomplished. The purpose of software compliance to DO-178B is to provide considerable confidence that the software is suitable for use in airborne systems. DO-178B should not be viewed as a documentation guide.

Compliance data are intended to be a consequence of the process. Complexity and extent of the required compliance data depend upon the characteristics of the system/software, associated development practices, and the interpretation of DO-178B, especially when it is applied to new technology and no precedent is available.

Finally, it has to be emphasized that DO-178B objectives do not directly deal with safety. Safety is dealt with at the system level via the system safety assessment. DO-178B objectives help verify the correct implementation of safety-related requirements that flow from the system safety assessment. Like any standard, DO-178B has good points and bad points (and even a few errors). However, careful consideration of its contents, taken together with solid engineering judgment, should result in better and safer airborne software.

References

1. RTCA DO-178B, *Software Considerations in Airborne Systems and Equipment Certification*, RTCA Inc., Washington, D.C, 1992. Copies of DO-178B may be obtained from RTCA, Inc., 1140 Connecticut Avenue, NW, Suite 1020, Washington, D.C. 20036-4001 U.S. (202) 833-9339. This document is also known as ED 12B, *Software Considerations in Airborne Systems and Equipment*

- Certification*, EUROCAE, Paris, 1992. Copies of ED-12B may be obtained from EUROCAE, 17, rue Hamelin, 75783 PARIS CEDEX France, (331) 4505-7188.
2. SAE ARP4754, *Certification Considerations for Highly-Integrated or Complex Aircraft Systems*, SAE, Warrendale, PA, 1996.
 3. Chilenski, J.J. and Miller, P.S., Applicability of modified condition/decision coverage to software testing, *Software Eng. J.*, 193, September 1994.
 4. Myers, G. J., *The Art of Software Testing*, John Wiley & Sons, New York, 1979.
 5. Beizer, B., *Software Testing Techniques*, 2nd ed., Coriolis Group, Scottsdale, AZ, 1990.
 6. McCracken, D. and Passafiume, M., *Software Testing and Evaluation*, Benjamin/Cummings, Menlo Park, CA, 1987.
 7. RTCA DO-248, Annual Report for Clarification of DO-178B “Software Considerations in Airbone Systems and Equipment Certification; EOROCAE ED-94, Annual Report for Clarification of ED-12B “Software Considerations in Airbone Systems and Equipment Certification.”

Further Information

1. The Federal Aviation Administration Web Page: www.faa.gov.
2. The RTCA Web Page: www.rtca.org.
3. Spitzer, C.R., *Digital Avionics Systems Principles and Practice*, 2nd ed., McGraw-Hill, New York, 1993.
4. Wichmann, B.A., A Review of a Safety-Critical Software Standard, National Physical Laboratory, Teddington, Middlesex, U.K. (report is not dated).
5. Herrman, D.S., *Software Safety and Reliability*, IEEE Computer Society Press, Washington, D.C., 1999.