

RTCA, Inc.
1150 18th Street, NW, Suite 910
Washington, D.C. 20036

Supporting Information for DO-178C and DO-278A

RTCA DO-248C
December 13, 2011

Prepared by: SC-205
© 2011 RTCA, Inc.

Copies of this document may be obtained from

RTCA, Inc.

Telephone: 202-833-9339

Facsimile: 202-833-9434

Internet: www.rtca.org

Please visit the RTCA Online Store for document pricing and ordering information.

FOREWORD

This document was prepared by RTCA Special Committee 205 (SC-205) and EUROCAE Working Group 71 (WG-71) and approved by the RTCA Program Management Committee (PMC) on December 13, 2011.

RTCA, Incorporated is a not-for-profit corporation formed to advance the art and science of aviation and aviation electronic systems for the benefit of the public. The organization functions as a Federal Advisory Committee and develops consensus-based recommendations on contemporary aviation issues. RTCA's objectives include but are not limited to:

- coalescing aviation system user and provider technical requirements in a manner that helps government and industry meet their mutual objectives and responsibilities;
- analyzing and recommending solutions to the system technical issues that aviation faces as it continues to pursue increased safety, system capacity, and efficiency;
- developing consensus on the application of pertinent technology to fulfill user and provider requirements, including development of minimum operational performance standards for electronic systems and equipment that support aviation; and
- assisting in developing the appropriate technical material upon which positions for the International Civil Aviation Organization and the International Telecommunication Union and other appropriate international organizations can be based.

The organization's recommendations are often used as the basis for government and private sector decisions as well as the foundation for many Federal Aviation Administration Technical Standard Orders.

Since the RTCA is not an official agency of the United States Government, its recommendations may not be regarded as statements of official government policy unless so enunciated by the U.S. government organization or agency having statutory jurisdiction over any matters to which the recommendations relate.

This Page Intentionally Left Blank

TABLE OF CONTENTS

1.0	INTRODUCTION	1
1.1	Purpose	1
1.2	Document Overview	2
1.3	How to Use This Document.....	2
2.0	ERRATA	3
3.0	FREQUENTLY ASKED QUESTIONS (FAQ)	5
3.1	FAQ #1: Section 2 of DO-178B provides an introduction to the system aspects relating to software development and notes that guidelines were under development at the time of writing. Where are these system life cycle guidelines documented?	5
3.2	FAQ #2: Throughout DO-178B reference is made to the system safety assessment process. Where can guidelines for this process be found?	5
3.3	FAQ #3: What is meant by safety monitoring software experiencing transients in DO-178B Section 2.3.2, paragraph 3?.....	5
3.4	FAQ #4: Does DO-178C/DO-278A’s definition of commercial off-the-shelf (COTS) software include COTS software designed for option-selectable software?	5
3.5	FAQ #5: What are “end-to-end checks” in the context of field-loadable software?	6
3.6	FAQ #6: What are the design description and verification activity objectives for a Level D system and why are there apparent inconsistencies in the objectives to be satisfied in Annex A?.....	6
3.7	FAQ #7: How can compliance with DO-178C/DO-278A section 5.2.3, Designing for User-Modifiable Software, be obtained?	7
3.8	FAQ #8: Can option-selectable software contain deactivated code?.....	7
3.9	FAQ #9: Do all high-level requirements require hardware/software integration testing? And, what does “ <i>To verify the interrelationships between software requirements and components</i> ” mean?	8
3.10	FAQ #10: Are baselines allowed to be changed? Section 7.2.2.c states baselines should be protected from change, whereas section 7.2.4.c talks about changes to baselines.....	8
3.11	FAQ #11: Is the "approved source" in section 7.2.7.a of DO-178B the previous approved product or is it the organization building the product?	8
3.12	FAQ #12: What are the definitions of Control Categories 1 and 2 (CC1 and CC2)?	8
3.13	FAQ #13: How is Table 7-1 in section 7.3 used to understand Control Categories 1 and 2 (CC1 and CC2)?.....	9
3.14	FAQ #14: What do Control Categories 1 and 2 (CC1 and CC2) mean when applied to the objectives of Annex A?	9
3.15	FAQ #15: Is software certified as a stand-alone product?	9

3.16	FAQ #16: What is the highest software level (per DO-178C) or assurance level (per DO-278A) that can be attained for previously developed software (PDS)?	10
3.17	FAQ #17: What are the issues related to changing previously developed software (PDS) versions from an earlier baseline?	10
3.18	FAQ #18: Since there is no specific guidance for handling changes to the aircraft’s operational environment, what part of DO-178C addresses this type of change?	11
3.19	FAQ #19: How does one determine if in-service problems indicate an inadequate process, and can one continue to pursue a service history means of compliance with some process inadequacies?	11
3.20	FAQ #20: What is the source of the glossary of terms in DO-178C, DO-278A, and the Supplements, and why do they appear to be different from other standard definitions?.....	12
3.21	FAQ #21: How is the second sentence of the definition of “patch” in DO-178C/DO-278A Annex B relevant to the definition itself?	12
3.22	FAQ #22: Can various industry process assessments such as the Software Engineering Institute (SEI) Capability Maturity Model Integration (CMMI), Software Process Improvement Capability Evaluation (SPICE), etc. be used for certification credit?	13
3.23	FAQ #23: Is software reliability addressed by DO-178C/DO-278A?.....	13
3.24	FAQ #24: What is the relationship between ARP4754A/ED-79A and DO-178C?.....	14
3.25	FAQ #25: Can architectural means be used to reduce the software level/assurance level needed for the incorporation of previously developed software (PDS) in a system?	15
3.26	FAQ #26: Does the fulfillment of “independence of multiple-version dissimilar software” (DO-178B section 12.3.3.1) supersede the independence requirements as defined in Annex A of DO-178B?.....	16
3.27	FAQ #27: What is meant by “user-modifiable software”?	16
3.28	FAQ #28: What is the value of removing <i>dead code</i> or <i>unused variables</i> ?	16
3.29	FAQ #29: What does DO-178C section 2.5.5.b (DO-278A section 2.6.5.b) mean, when it addresses requirements related to a default mode, if one is provided to protect against software load errors?	16
3.30	FAQ #30: What does DO-178B section 2.6a(2) mean regarding system safety requirements addressing system anomalous behavior?	16
3.31	FAQ #31: How does verification of product relate to “compiler acceptability”?	16
3.32	FAQ #32: What are defensive programming practices?	17
3.33	FAQ #33: Is it permissible to NOT meet the safety objectives by justifying any deviations from the design standards?	19
3.34	FAQ #34: What is the concept of independence as used in DO-178B?	19
3.35	FAQ #35: What are low-level requirements and how may they be tested?.....	19
3.36	FAQ #36: What is the definition or interpretation of derived requirements in DO-178C/DO-278A?.....	20

3.37	FAQ #37: What is meant by providing derived software requirements to the system processes, including the system safety assessment process?	20
3.38	FAQ #38: What is the difference between Integration Process and Integration Testing?	21
3.39	FAQ #39: What is the definition of an unbounded recursive algorithm in DO-178C/DO-278A section 6.3.3.d?	21
3.40	FAQ #40: What representation of the software is used to perform reviews and analyses of Source Code?	21
3.41	FAQ #41: Why is Source Code to object code traceability required for Level A software?.....	22
3.42	FAQ #42: What needs to be considered when performing structural coverage at the object code level?	22
3.43	FAQ #43: What is the intent of structural coverage analysis?.....	23
3.44	FAQ #44: Why is structural testing not a DO-178C/DO-278A requirement?	24
3.45	FAQ #45: What is the relevance of the exception case stated in the definition of dead code?.....	24
3.46	FAQ #46: What is the meaning of section 7.2.2.g in DO-178C/DO-278A, when it states that a baseline or configuration item should be traceable either to the output it identifies or to the process with which it is associated?	25
3.47	FAQ #47: What is meant by the term “certification credit” or “approval credit”?.....	25
3.48	FAQ #48: In addition to sections 9 and 10 of DO-178C, which provide a high-level overview of the certification liaison and certification processes, where can further guidance on the approval process for software be found?....	26
3.49	FAQ #49: Where can current certification authority guidance regarding issues not covered in DO-178B or expanding upon issues in DO-178B be found?	27
3.50	FAQ #50: What data items are deliverable to the certification/approval authority to support software approval and product certification/approval?	27
3.51	FAQ #51: What is meant by the term “type design,” as used in section 9.4 of DO-178B?	27
3.52	FAQ #52: Why do the certification/approval authorities not approve an organization’s process once, rather than approve each product submitted as part of a certification/approval application?	27
3.53	FAQ #53: Do the data items need to be prepared and packaged as specified in section 11 of DO-178B?	28
3.54	FAQ #54: Is the documentation required in DO-178C/DO-278A section 11 excessive, especially for small projects?	28
3.55	FAQ #55: What are the control category considerations when determining how to package the data items discussed in section 11 of DO-178C/DO-278A?.....	28
3.56	FAQ #56: How are redundancies inherent in the software verification documents eliminated?	29

3.57	FAQ #57: Is it necessary to mention all the additional considerations in the Plan for Software Aspects of Certification (PSAC) or Plan for Software Aspects of Approval (PSAA) or is it sufficient to mention only the applicable additional considerations?	29
3.58	FAQ #58: How do you implement re-verification?	29
3.59	FAQ #59: What type of non-flight software is covered by DO-178C?	30
3.60	FAQ #60: Is a complete set of plans required for modifications of a system?	30
3.61	FAQ #61: What constitutes a development tool and when should it be qualified?	31
3.62	FAQ #62: What are the requirements for flight test analysis software and ground-based test software?.....	31
3.63	FAQ #63: For exhaustive input testing, the applicant should provide an analysis which confirms the isolation of the inputs to the software. What does it mean to confirm the isolation?	32
3.64	FAQ #64: Is it sufficient to use different linker or loader to produce dissimilar versions for avionics software?	32
3.65	FAQ #65: What is meant by “ <i>equivalent software verification process activity</i> ” in DO-178C/DO-278A sections 12.3.2.4 (Tool Qualification for Multiple-Version Dissimilar Software) and 12.3.2.5 (Multiple Simulators and Verification)?	33
3.66	FAQ #66: What is the difference between certification, approval, and qualification?	33
3.67	FAQ #67: What is analysis of data coupling and control coupling?	33
3.68	FAQ #68: The third sentence of the third paragraph of section 3.2 of DO-178C states that “ <i>Component X illustrates the use of previously developed software used in a certified product.</i> ” (The equivalent sentence in DO-278A states that “ <i>Component X illustrates the use of previously developed software used in an approved system.</i> ”) Is it necessary, for a reused component to have been used in the context of a previously certified product or an approved system?	35
3.69	FAQ #69: What is the rationale to have software design process feedback to the planning process in section 5.2.2.g of DO-178C/DO-278A, where feedback to the system life cycle process and software requirements process seems adequate?	35
3.70	FAQ #70: What is the purpose of the second sentence in DO-178C/DO-278A section 5.2.4.b?	36
3.71	FAQ #71: What is the purpose of traceability, how much is required, and how is it documented? For example, is a matrix required or are other methods acceptable?	36
3.72	FAQ #72: What happens if an error indicates a weakness in the development process itself?	36
3.73	FAQ #73: Are timing measurements during testing sufficient or is a rigorous demonstration of worst-case timing necessary?	37

3.74	FAQ #74: What is the difference between the development and life cycle objectives stated in DO-178C for Level A versus Level B software, and how does that relate to safety? Similarly, what is the difference in DO-278A for AL1 versus AL2 software?.....	38
3.75	FAQ #75: Can sampling be used for some verification activities (such as coding rules on Source Code)?	39
3.76	FAQ #76: Can Problem Reports and verification activities performed on a software configuration item be referenced in previously approved products without repeating this effort for each product that uses this software configuration item?	40
3.77	FAQ #77: The Software Requirements Data are described by DO-178C/DO-278A section 11.9. What is meant in step 11.9.g: “ <i>Failure detection and safety monitoring requirements</i> ”?	40
3.78	FAQ #78: For software requirements expressed by logic equations, how many normal range test cases are necessary to verify the variable usage and the Boolean operators?.....	41
3.79	FAQ #79: Can an applicant for aircraft, engine, or propeller certification take credit for DO-178C compliance found under an article approval (that is, Technical Standard Order (TSO) or European Technical Standard Order (ETSO))?.....	41
3.80	FAQ #80: What needs to be considered when using inlining?	42
3.81	FAQ #81: What aspects should be considered when there is only one level of requirements (or if high-level requirements and low-level requirements are merged)?	43
3.82	FAQ #82: If pseudocode is used as part of the low-level requirements, what issues need to be addressed?	44
3.83	FAQ #83: Should compiler errata be considered?.....	46
3.84	FAQ #84: How can all Level D (AL 5) objectives be met if low-level requirements and Source Code are not required?	46
4.0	DISCUSSION PAPERS (DP).....	49
4.1	DP #1: Verification Tool Selection Considerations.....	49
4.2	DP #2: The Relationship of DO-178B/ED-12B to the Code of Federal Regulations (CFRs) and Joint Aviation Requirements (JARs).....	49
4.3	DP #3: The Differences Between DO-178A and DO-178B Guidance for Meeting the Objective of Structural Coverage	49
4.4	DP #4: Service History Rationale for DO-178C.....	49
4.5	DP #5: Application of Potential Alternative Methods of Compliance for Previously Developed Software (PDS).....	55
4.6	DP #6: Transition Criteria.....	67
4.7	DP #7: Definition of Commonly Used Verification Terms	70
4.8	DP #8: Structural Coverage and Safety Objectives	70
4.9	DP #9: Assessment and Classification of Open Software Problems	71
4.10	DP #10: Considerations Addressed When Deciding to Use Previously Developed Software (PDS).....	76

4.11	DP #11: Qualification of a Tool Using Service History	79
4.12	DP #12: Object Code to Source Code Traceability Issues.....	79
4.13	DP #13: Discussion of Statement Coverage, Decision Coverage, and Modified Condition/Decision Coverage (MC/DC)	81
4.14	DP #14: Partitioning Aspects in DO-178C/DO-278A.....	88
4.15	DP #15: Relationship Between Regression Testing and Hardware Changes ...	92
4.16	DP #16: Cache Management	94
4.17	DP #17: Usage of Floating-Point Arithmetic.....	96
4.18	DP #18: Service Experience Rationale for DO-278A	97
4.19	DP #19: Independence in DO-178C/DO-278A	103
4.20	DP #20: Parameter Data Items and Adaptation Data Items.....	109
4.21	DP #21: Clarification on Single Event Upset (SEU) as It Relates to Software.....	118
5.0	RATIONALE FOR DO-178C/DO-278A	125
5.1	Introduction.....	125
5.2	Rationale for DO-178C/DO-278A Section 2: SYSTEM ASPECTS RELATING TO SOFTWARE DEVELOPMENT	126
5.3	Rationale for DO-178C/DO-278A Section 3: SOFTWARE LIFE CYCLE....	126
5.4	Rationale for DO-178C/DO-278A Section 4: SOFTWARE PLANNING PROCESS	126
5.5	Rationale for DO-178C/DO-278A Section 5: SOFTWARE DEVELOPMENT PROCESSES	127
5.6	Rationale for DO-178C/DO-278A Section 6: SOFTWARE VERIFICATION PROCESS	128
5.7	Rationale for DO-178C/DO-278A Section 7: SOFTWARE CONFIGURATION MANAGEMENT PROCESS.....	130
5.8	Rationale for DO-178C/DO-278A Section 8: SOFTWARE QUALITY ASSURANCE PROCESS	130
5.9	Rationale for DO-178C/DO-278A Section 9: CERTIFICATION/APPROVAL LIAISON PROCESS	131
5.10	Rationale for DO-178C/DO-278A Section 10: OVERVIEW OF CERTIFICATION (CNS/ATM SYSTEM APPROVAL) PROCESS.....	131
5.11	Rationale for DO-178C/DO-278A Section 11: SOFTWARE LIFE CYCLE DATA	131
5.12	Rationale for DO-178C/DO-278A Section 12: ADDITIONAL CONSIDERATIONS.....	131
5.13	Rationale for TOOL QUALIFICATION DOCUMENT AND DO- 178C/DO-278A SUPPLEMENTS.....	132

APPENDIX A – ACRONYMS	A-1
APPENDIX B – COMMITTEE MEMBERSHIP	B-1
APPENDIX C – INDEX OF KEYWORDS	C-1
APPENDIX D – CORRELATION BETWEEN DO-178C, DO-278A, AND DO-248C	D-1

TABLE OF FIGURES

FIGURE 4-1 PDS RELATIONSHIP TO PSAC/PSAA.....	58
FIGURE 4-2 INDEPENDENCE AND VERIFICATION OBJECTIVES ILLUSTRATED	106
FIGURE 4-3 PARAMETER DATA ITEM RELATIONSHIP WITH EXECUTABLE OBJECT CODE	110
FIGURE 4-4 PDI VIEW IN DO-178C AND DO-278A.....	111
FIGURE 4-5 EXAMPLE OF PDI HIGH-LEVEL REQUIREMENT	112
FIGURE 4-6 EXAMPLE OF PDI HIGH-LEVEL REQUIREMENTS.....	113
FIGURE 4-7 EXAMPLE OF HIGH-LEVEL REQUIREMENT DEFINING PDI VALUES.....	113
FIGURE 4-8 VERIFICATION OF PDI FILE	116

TABLE OF TABLES

TABLE 4-1 TRUTH TABLE FOR DECISION (A OR (B AND C))	85
TABLE 4-2 TRUTH TABLE FOR DECISIONS (A (B && C))	86
TABLE 4-3 TRUTH TABLE FOR ((A B) && C)	87
TABLE 4-4 TRUTH TABLE FOR ((A < 3) OR (A > 8))	87
TABLE 4-5 INDEPENDENCE INTERPRETATION FOR VERIFICATION OBJECTIVES	107
TABLE 4-6 DATA TYPE ATTRIBUTES	114

This Page Intentionally Left Blank

1.0 INTRODUCTION

DO-178C, “Software Considerations in Airborne Systems and Equipment Certification,” provides recommendations for the production of software for airborne systems and equipment that performs its intended function with a level of confidence in safety that complies with airworthiness requirements. DO-278A, “Software Integrity Assurance Considerations for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems,” provides similar guidance for the CNS/ATM ground-based community. DO-178C and DO-278A are based on DO-178B; therefore, many of the questions that were raised by the aviation community against DO-178B are also relevant to DO-178C and DO-278A.

This document addresses the questions of both the industry and authorities. It contains frequently asked questions (FAQs), discussion papers (DPs), and rationale. Many of the FAQs and DPs are based on DO-248B; however, some have been modified to address changes from DO-178B to DO-178C and to make applicable to DO-278A. Additionally, some new FAQs and DPs have been added to provide additional clarification on DO-178C and/or DO-278A. The errata against DO-178B (which were in section 2 of DO-248B) have been deleted, since they have been incorporated into DO-178C and are no longer relevant. The rationale for DO-178C and DO-278A objectives has also been added to DO-248C.

Note: Prior to using this document, it is recommended that the reader consider section 1.3, “How to Use This Document”.

1.1 Purpose

This document provides clarification of the guidance material in DO-178C and DO-278A. In order to accomplish this clarification the following products have been generated:

- Frequently Asked Question (FAQ) – Section 3: The purpose of a FAQ is to provide short and concise responses to questions that are frequently asked by industry concerning the material of DO-178C and/or DO-278A. These questions are frequently posed to certification authorities or others who provide interpretation of DO-178C and/or DO-278A. A FAQ contains no new guidance material. A FAQ is typically no longer than two pages.
- Discussion Paper (DP) – Section 4: The purpose of a Discussion Paper is to provide clarification for certain sections of DO-178C and/or DO-278A in cases where the clarification requires more than a short answer to a question. A DP contains no new guidance material.
- Rationale – Section 5: The purpose of the rationale is to document some items considered when developing DO-178B and then DO-178C and DO-278A. The rationale is only intended to be background information to aid the reader’s understanding of DO-178C and DO-278A — especially of those sections and objectives that industry feedback has shown might benefit from additional clarification. The rationale contains no guidance material.

1.2 Document Overview

The products approved by RTCA Special Committee #205 (SC-205) and EUROCAE Working Group #71 (WG-71) are included in this document as follows:

- Section 3 includes the FAQs.
- Section 4 includes the DPs.
- Section 5 provides the rationale.
- Appendix A includes a list of all acronyms that are used throughout this document.
- Appendix B includes a list of the SC-205/WG-71 committee membership.
- Appendix C includes an index of keywords used in this document. The user may utilize this index to find information on related topics.
- Appendix D contains correlation table between DO-248C, DO-178C, and DO-278A.

1.3 How to Use This Document

Sections 3 and 4 are essentially chronological. DO-248 started out as an annual report, which was finalized with the publication of DO-248B. In order to keep the numbering for FAQs and DPs consistent with DO-248B, new FAQs were added to the end of section 3 and new DPs were added to the end of section 4. Several FAQs and DPs were deleted because the information is now clarified in DO-178C/DO-278A, in another FAQ or DP, or in one of the DO-178C/DO-278A supplements. The original titles of these FAQs and DPs are maintained for historical purposes. The rationale in section 5 is new to DO-248C. Because of the document structure, it is not recommended to read it from front to back. Instead the following approaches are recommended for using the document:

- Topical approach: If you are researching a specific topic, Appendix C provides an index of keywords. Each keyword is shown with the multiple sections that use that keyword.
- Reference approach: If you are researching a specific section in DO-178C/DO-278A, use Appendix D. Appendix D provides a mapping of the DO-178C/DO-278A sections to the related FAQs or DPs in this document.
- Hyperlink approach: Hyperlinks are provided from Appendices C and D to the FAQs and DPs in the main body in electronic versions of this document. You may use these hyperlinks to immediately go to the referenced section.

Major sections are numbered as X.0 throughout this document. It should be noted that references to an entire section are identified as “section X”; whereas, references to the content between section headers X.0 and X.1 are referenced as “section X.0”.

2.0 ERRATA

At the time of this document's publication, no errata for DO-178C or DO-278A are known.

This Page Intentionally Left Blank

3.0 FREQUENTLY ASKED QUESTIONS (FAQ)

This section compiles the DO-178C and DO-278A FAQs. The purpose of a FAQ is to provide short and concise responses to questions that are frequently asked by industry concerning the material of DO-178C and/or DO-278A. These questions are frequently posed to certification authorities or others who provide interpretation of DO-178C and/or DO-278A. A FAQ contains no guidance material.

Disclaimer: The responses to FAQs have been prepared and approved by Joint Committee SC-205/WG-71. These FAQs have no recognition by the certification authorities and are provided for information purposes only.

3.1 **FAQ #1: Section 2 of DO-178B provides an introduction to the system aspects relating to software development and notes that guidelines were under development at the time of writing. Where are these system life cycle guidelines documented?**

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.2 **FAQ #2: Throughout DO-178B reference is made to the system safety assessment process. Where can guidelines for this process be found?**

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.3 **FAQ #3: What is meant by safety monitoring software experiencing transients in DO-178B Section 2.3.2, paragraph 3?**

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.4 **FAQ #4: Does DO-178C/DO-278A's definition of commercial off-the-shelf (COTS) software include COTS software designed for option-selectable software?**

Reference: DO-178C: Sections 2.5.4, 4.2.h, 5.2.4, and 6.4.4.3.d
DO-278A: Sections 2.6.4, 4.2.h, 5.2.4, and 6.4.4.3.d

Keywords: commercial off-the-shelf software; COTS; option-selectable software

Answer:

The DO-178C glossary defines COTS software as: *“Commercially available applications sold by vendors through public catalog listings. COTS software is not intended to be customized or enhanced. Contract-negotiated software developed for a specific application is not COTS software.”*

DO-278A glossary defines COTS software slightly different from DO-178C to add clarification for CNS/ATM systems: *“Software under consideration for use in a CNS/ATM system that may have no or only partial evidence of compliance to this document sections 4 – 9 objectives.”*

As long as the COTS software is not changed as supplied from the vendor, it is still considered COTS software per the definitions above.

If the COTS software has option-selectable functionality, some deactivated code may exist. For an explanation of how to address deactivated code, see the DO-178C or DO-278A sections referenced above.

For all products, where the capability for the user to select options or functions is not provided by the vendor, and where the customer has to change the software, one needs to consider this software as previously developed software. This will not change the objectives to be fulfilled for certification/approval.

3.5 **FAQ #5: What are “end-to-end checks” in the context of field-loadable software?**

Reference: DO-178C: Section 2.5.5.d
DO-278A: Section 2.6.5.d

Keywords: end-to-end checks; field-loadable software

Answer:

“End-to-end” checking means that the entire software load is verified – bit by bit. It starts with the production of the load media and ends with verification of the load being successfully completed. This includes the confirmation that displayed part numbers, etc. are shown to be correct.

Software that performs the loading, or integrity checking of the load, of field-loadable application software is typically operational when the system is not operational, for example, on the ground.

Proper system operation after the load is dependent on a correct load of the application software and the ability to verify the configuration (for example, a review of part numbers by maintenance personnel).

Integrity of the load can sometimes be assured without verification of the complex software responsible for this loading and checking to the same level as the software being loaded. This can be done via simple checks that are independent of the load checking software. These checks should be developed and verified to the level of the loaded software. These simple checks, referred to as end-to-end checks, generally use memory checksum, read/write verification, or cyclic redundancy checks (CRC) to confirm load correctness. Such checks are independent from the software that actually performed the load and from any associated part number display or checks.

3.6 **FAQ #6: What are the design description and verification activity objectives for a Level D system and why are there apparent inconsistencies in the objectives to be satisfied in Annex A?**

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.7 **FAQ #7: How can compliance with DO-178C/DO-278A section 5.2.3, Designing for User-Modifiable Software, be obtained?**

Reference: DO-178C/DO-278A: Section 5.2.3

Keywords: non-modifiable software; user-modifiable software

Answer:

DO-178C/DO-278A section 5.2.3.b states: *“The means provided to change the modifiable component should be shown to be the only means by which the modifiable component can be changed.”*

The assumption is made from the outset that compliance with DO-178C/DO-278A section 5.2.3.a is in place; that is, that the non-modifiable software is completely protected from interference from the user-modifiable portion, and that the conditions of DO-178C/DO-278A 5.2.3.a are also met. Further, the effects of the user-modifiable software should be shown to have no adverse effect on the overall system safety through the protection mechanisms employed, for example, during loading.

The issue now becomes one of how the change to the user-modifiable software is implemented. This is established through the applicant's defined means. The means can be represented in various ways (for example, defined tools, methodology, or rules). As long as the user adheres to these means, only the intended modification will result. For example, the applicant may define a set of design requirements for the user-modifiable component that is specific enough that, along with a defined implementation methodology, it constitutes the only means to implement the change. The applicant should show that the protection mechanisms around the modifiable component shield the non-modifiable component from unexpected changes. Caution should be exercised that a change in the user-modifiable software does not trigger a change in the non-modifiable software.

3.8 **FAQ #8: Can option-selectable software contain deactivated code?**

Reference: DO-178C: Sections 2.5.4, 4.2.h, 5.2.4, and 6.4.4.3.d
DO-278A: Sections 2.6.4, 4.2.h, 5.2.4, and 6.4.4.3.d

Keywords: deactivated code; option-selectable software

Answer:

Yes, please refer to DO-178C section 2.5.4 (DO-278A section 2.6.4) for details for description of option-selectable software, and DO-178C/DO-278A sections 4.2.h and 5.2.4 for designing for deactivated code.

3.9 FAQ #9: Do all high-level requirements require hardware/software integration testing? And, what does “To verify the interrelationships between software requirements and components” mean?

Reference: DO-178C/DO-278A: Sections 6.4, 6.4.1, and 6.4.3

Keywords: hardware/software integration test; high-level requirements; integration; interrelationships; software integration test

Answer:

DO-178C/DO-278A section 6.4 defines hardware/software integration testing as a process that verifies the “*correct operation of the software in the target computer environment.*”

DO-178C/DO-278A section 6.4.3.a states: “*Requirements-based hardware/software integration testing ensures that the software in the target computer will satisfy the high-level requirements.*”

So, the answer to the first question is “no”: It is not required that all high-level requirements testing be executed on the target computer. Reference DO-178C/DO-278A section 6.4.1 that states that “*more than one test environment may be needed to satisfy the objectives for software testing.*” However, testing conducted on a host computer (non-target computer environment) may need justification for their results validity for the target computer system.

For the second question, see DO-178C/DO-278A section 6.4.3.b, which states: “*Requirements-based software integration testing ensures that the software components interact correctly with each other and satisfy the software requirements and software architecture.*”

3.10 FAQ #10: Are baselines allowed to be changed? Section 7.2.2.c states baselines should be protected from change, whereas section 7.2.4.c talks about changes to baselines.

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.11 FAQ #11: Is the “approved source” in section 7.2.7.a of DO-178B the previous approved product or is it the organization building the product?

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.12 FAQ #12: What are the definitions of Control Categories 1 and 2 (CC1 and CC2)?

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.13 FAQ #13: How is Table 7-1 in section 7.3 used to understand Control Categories 1 and 2 (CC1 and CC2)?

Reference: DO-178C/DO-278A: Section 7.3

Keywords: configuration management; control categories; Control Category 1; CC1; Control Category 2; CC2

Answer:

DO-178C/DO-278A section 7.3 defines the configuration management objectives that should be satisfied for CC1 and for CC2. The entries in column 2 of table 7-1 are references to all the objectives of SCM that apply to software life cycle data, with the exceptions of Software Load Control and Software Life Cycle Environment Control. The black dots in column 3 of the table indicate that all the SCM objectives apply for data which should meet CC1 objectives. The black dots in column 4 indicate the subset of SCM objectives that apply for data which should meet CC2 objectives.

3.14 FAQ #14: What do Control Categories 1 and 2 (CC1 and CC2) mean when applied to the objectives of Annex A?

Reference: DO-178C/DO-278A: Section 7.3 and Annex A

Keywords: configuration management; data control categories; Control Category 1; CC1; Control Category 2; CC2

Answer:

CC1 and CC2 define the processes and activities for handling the data that is generated to show that each Annex A objective has been achieved. CC1 and CC2 address configuration management (CM) issues. These issues are handled in different ways depending on the type of data.

The tables in Annex A of DO-178C/DO-278A specify which data control category (CC1 or CC2) applies by software level/assurance level for each item of software life cycle data.

The last two columns of DO-178C/DO-278A Table 7-1 define which objectives are applicable to CC1 and which objectives are applicable to CC2. In general, CC2 is a simpler method of control and is applied to data items based on the type of data and the software level/assurance level. Annex A defines the categorization for each type of data.

Not all data needs to be controlled using the same method to meet the CC1 and CC2 objectives. For example, Software Quality Assurance (SQA) Records may be controlled by placing them in a development library, whereas the product executable code may be controlled by an independent CM organization.

3.15 FAQ #15: Is software certified as a stand-alone product?

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.16 FAQ #16: What is the highest software level (per DO-178C) or assurance level (per DO-278A) that can be attained for previously developed software (PDS)?

Reference: DO-178C/DO-278A: Section 12.1

Keywords: assurance level; commercial off-the-shelf software; COTS; previously developed software; PDS; software level

Answer:

The highest software level/assurance level obtainable for previously developed software (PDS) can be Level A/AL1. For this to be achieved, the PDS should satisfy all the applicable objectives of DO-178C/DO-278A.

Business considerations of using PDS could prevent an applicant from reaching the highest level that is technically possible for that software.

3.17 FAQ #17: What are the issues related to changing previously developed software (PDS) versions from an earlier baseline?

Reference: DO-178C/DO-278A: Sections 7.2.2, 7.2.4, and 12.1

Keywords: baseline; change; commercial off-the-shelf software; COTS; modification; previously developed software; PDS

Answer:

For this FAQ, commercial off-the-shelf (COTS) software is considered a subset of PDS.

The system with PDS in question may have been certified or approved using means other than DO-178C/DO-278A. Regulatory information exists to provide guidance for approval for software changes in legacy systems using DO-178C/DO-278A.

The effort to incorporate the changed PDS baseline should follow the same process as incorporating changes in any software product per the objectives of DO-178C/DO-278A. This includes, but is not limited to, an evaluation of the:

- Impact of PDS requirements changes on system requirements.
- Impact of the change on the system safety analysis.
- Impact of the change on the previously accepted certification data package to determine the scope of the change and to decide if the change should or can be accomplished.
- Impact of changes on software life cycle data.
- Impact to re-verification efforts.
- Process effort to implement and verify the PDS modification (if selected) into the system.

The assumption is made that the PDS has been previously incorporated into a certified system. The decision to change the baseline version of PDS is based on the same considerations used in the original selection of the PDS. A new accept or reject decision should be made before going forward with the change. It is important to gain access to the provider's Problem Reports that were incorporated into the change, as well as the revised software requirements (new functionality) as a roadmap to incorporation and re-verification of the new version of the system software. It is important to capture any information related to problems fixed in between versions. Any life cycle data pertaining to the PDS should be obtained. The Problem Reports may be provided, even for COTS software. The revised software requirements statement may be available through a list of new features or an updated user's manual.

Configuration considerations for PDS should ensure baseline and subsequent updates are controlled by the applicant. For example, all received copies of PDS with the same identification may not be identical, yet all delivered copies from the applicant should be identical.

Further information on PDS may be obtained using the keyword index (Appendix C).

3.18 FAQ #18: Since there is no specific guidance for handling changes to the aircraft's operational environment, what part of DO-178C addresses this type of change?

Reference: DO-178C: Sections 12.1.1 and 12.1.2

Keywords: aircraft installation; change; modification; operational environment; previously developed software; PDS

Answer:

DO-178C section 12.1.1 addresses modifications to previously developed software (PDS). DO-178C section 12.1.2 addresses the case when the aircraft installation changes. While a change in an aircraft's operating environment (for example, maximum altitude or maximum airspeed) is not strictly a change in the aircraft installation, DO-178C section 12.1.2 can be used to cover a change to the operating environment of an existing aircraft installation. Specifically, DO-178C section 12.1.2.b covers the functional modifications to the aircraft which may include changes to software requirements due to changes to the operating environment. When using s DO-178C section 12.1.2, it should be assumed that the term "aircraft installation" includes both the installation and the environment.

3.19 FAQ #19: How does one determine if in-service problems indicate an inadequate process, and can one continue to pursue a service history means of compliance with some process inadequacies?

This FAQ has been removed, as approved by the SC-205/WG-71 committee. The information has been incorporated into DP #4.

3.20 FAQ #20: What is the source of the glossary of terms in DO-178C, DO-278A, and the Supplements, and why do they appear to be different from other standard definitions?

Reference: DO-178C/DO-278A: Annex B and Section 1.4

Keywords: definitions; glossary; terms

Answer:

Where possible, DO-178C/DO-278A uses standard definitions for the terms used in the document. These definitions are defined in industry standards such as Institute of Electrical and Electronic Engineers (IEEE) and Society of Automotive Engineers (SAE). Additionally, terms from European Aviation Safety Agency (EASA) and Federal Aviation Administration (FAA) documents are used.

The DO-178C/DO-278A glossary (Annex B) defines some terms more precisely or narrowly, where a special interpretation is required by the aviation community and in order to prevent ambiguity during the interpretation of the document. However, it was a goal of the joint committee developing the document to minimize the use of special terms or unique definitions in DO-178C. The DO-278A glossary stands alone from DO-178C (that is, DO-278A has its own complete glossary). It started with the DO-178C glossary and modified or added terms to address CNS/ATM non-airborne software development.

The DO-178C/DO-278A supplements also include glossaries. These glossaries contain terms which are unique to the technology discussed in the specific supplement. For terms that are not technology specific, the DO-178C/DO-278A glossary serves as the glossary for the supplements.

3.21 FAQ #21: How is the second sentence of the definition of “patch” in DO-178C/DO-278A Annex B relevant to the definition itself?

Reference: DO-178C/DO-278A: Annex B

Keywords: embedded identifiers; modification; patch

Answer:

DO-178C/DO-278A Annex B defines “patch” as: *“Modification to Executable Object Code, in which one or more of the planned steps of re-compiling, re-assembling, or re-linking is bypassed. This does not include embedded identifiers.”*

The second sentence of this definition is intended to show recognized exceptions to the normal case. Typically, embedded identifiers are values such as checksums or part numbers. In cases where the values cannot be determined at compile time, memory is allocated and values are inserted during or after linking. This type of modification is not considered a patch since it is not modifying the executable code. The second sentence provides this distinction.

3.22 FAQ #22: Can various industry process assessments such as the Software Engineering Institute (SEI) Capability Maturity Model Integration (CMMI), Software Process Improvement Capability Evaluation (SPICE), etc. be used for certification credit?

Reference: DO-178C/DO-278A: Sections: 1.3, 9.1, 11, and Annex A

Keywords: Capability Maturity Model Integration; CMMI; process assessment; Software Engineering Institute; SEI; Software Process Improvement Capability Evaluation; SPICE

Answer:

The simple answer is that an applicant cannot claim certification “credit” for the use of the SEI CMMI, SPICE, etc.

Although DO-178C/DO-278A and its predecessors advocate good process characteristics, process objectives are given primarily in terms of attributes of the process data and the associated product. Many of the DO-178C/DO-278A objectives specifically relate to completeness and rigor of development and verification processes in specific terms, whereas the generalized process models, such as the SEI CMMI and SPICE, are more focused on business process measurement and improvement. They stipulate and assess against many process attributes that are necessary, but not sufficient conditions, for high product integrity and safety. For example, the SEI CMMI has no process activities or capabilities or metrics associated with verification (review, analysis, or test) coverage, whereas this is a very significant aspect of DO-178C/DO-278A guidance, because verification is a key factor in system safety.

Although there is some overlap, DO-178C/DO-278A remains the more comprehensive guidance for airborne and CNS/ATM systems. Applicants seeking process recognition for the use of the SEI CMMI, SPICE, etc. may readily reuse the same process compliance data to substantiate compliance with DO-178C/DO-278A objectives. Such data would still be required to meet the software life cycle data attributes of DO-178C/DO-278A section 11 and the objectives of DO-178C/DO-278A Annex A with respect to independence and configuration control.

3.23 FAQ #23: Is software reliability addressed by DO-178C/DO-278A?

Reference: DO-178C/DO-278A: Sections 2.3 and 12.3.3

Keywords: software reliability

Answer:

DO-178C/DO-278A does mention software reliability; however, the concept of using numerical reliability figures for software is dismissed in DO-178C/DO-278A by making the following observations:

- DO-178C/DO-278A section 2.3 states that a software level/assurance level cannot be interpreted as a failure rate.
- DO-178C/DO-278A section 12.3.3 indicates that software reliability analysis based on development metrics does not provide sufficient confidence.

3.24 FAQ #24: What is the relationship between ARP4754A/ED-79A and DO-178C?

Reference: DO-178C: Section 2

Keywords: ARP4754A; ED-79A; hardware development; system development processes; system safety assessment; SSA; system safety assessment process

Answer:

Society of Automotive Engineers (SAE) Aerospace Recommended Practice ARP4754A or EUROCAE document ED-79A, “Guidelines for Development of Civil Aircraft and Systems,” was produced to provide guidance at the system level for systems and equipment that implement aircraft-level functions and to provide links to specific related disciplines, such as the software development life cycle. The relationships between the system safety assessment process, hardware development process, and software development process are discussed in DO-178C section 2. The following diagram illustrates these relationships.

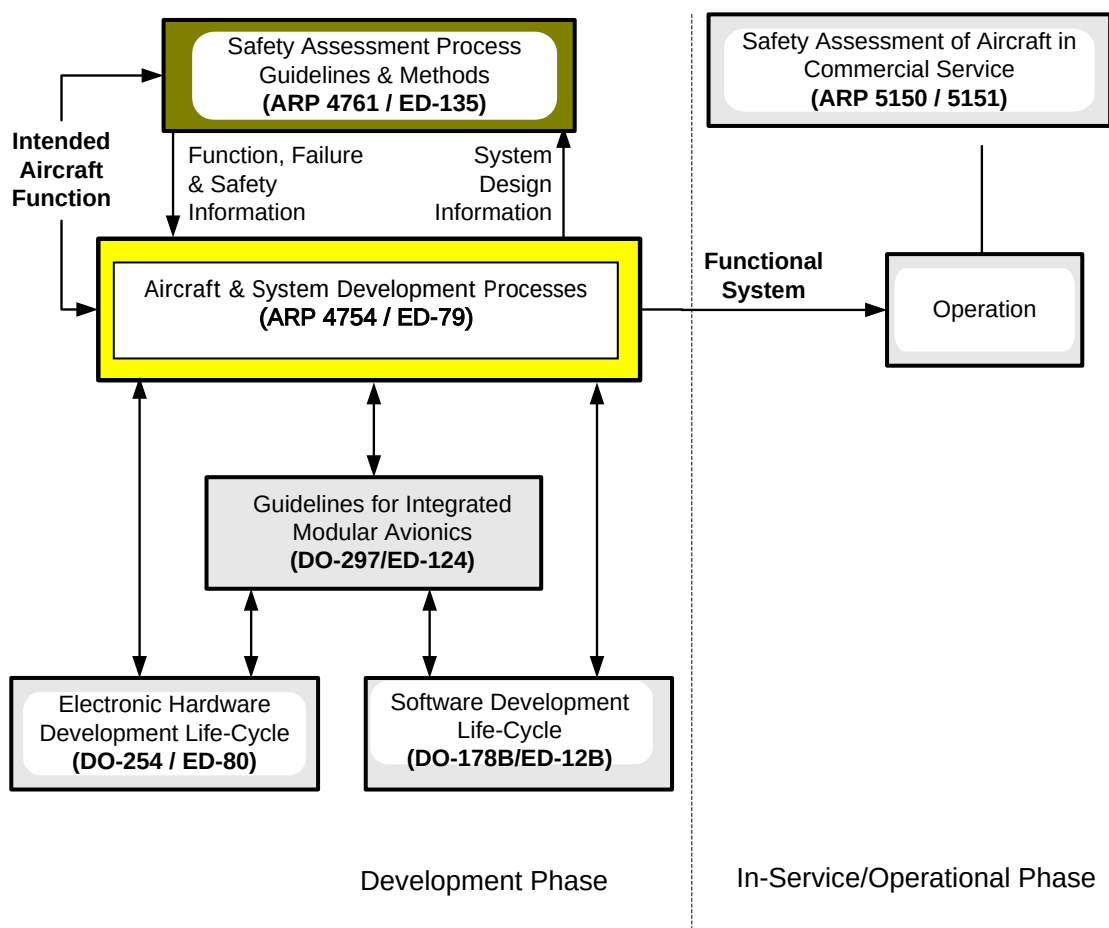


Figure 3-1 ARP4754/ED-79 Relationship To Other Documents

3.25 FAQ #25: Can architectural means be used to reduce the software level/assurance level needed for the incorporation of previously developed software (PDS) in a system?

Reference: DO-178C/DO-278A: Section 2.4

Keywords: architectural means; architecture; assurance level; commercial off-the-shelf; COTS; partitioning; previously developed software; PDS; software level

Answer:

Yes, architectural means can be used to reduce the software level/assurance level needed by mitigating all identified risks that could result from anomalous behavior of the PDS. The methods discussed in DO-178C/DO-278A section 2.4 are applicable to all types of software including PDS. The required justification associated with these methods (such as analysis with respect to design independence, etc.) is also the same for PDS as for any other software. If sufficient justification data cannot be obtained, this may limit the extent to which the architectural means may be used. Methods for implementing architectural means discussed in SAE ARP4754A (and EUROCAE ED-79A) and DO-178C/DO-278A are also potentially applicable to PDS. Some of the architectural means referenced in these documents are listed below:

- Partitioning.
- Safety monitoring.
- Multiple-version dissimilar software (with use of monitors, comparators, and polling) may be used for redundancy or backup.

Assurance needed to implement these methods is also discussed in DO-178C/DO-278A section 2.4. Further considerations in the implementation of these methods include but are not restricted to:

- Whether the resulting function is a primary or a secondary function.
- Probability of common mode failures.
- Analysis of common cause errors.

Depending upon the resultant architecture, more assurance of the architectural means may be needed than that of the PDS.

Note also that restriction of functionality may be used in the case of PDS. In general, many PDS products include a variety of functions that might not be needed for any given application and thus it might be necessary to subject all of them to the same degree of scrutiny unless unneeded functions are restricted. There should be an analysis provided to show why and how unneeded code is deactivated. Restriction of functionality may result in deactivated code in the resultant PDS component. Additionally, PDS may also contain extraneous or dead code. All objectives regarding deactivated, extraneous, and dead code also apply to PDS.

- 3.26 FAQ #26: Does the fulfillment of “independence of multiple-version dissimilar software” (DO-178B section 12.3.3.1) supersede the independence requirements as defined in Annex A of DO-178B?**

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

- 3.27 FAQ #27: What is meant by “user-modifiable software”?**

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

- 3.28 FAQ #28: What is the value of removing *dead code* or *unused variables*?**

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

- 3.29 FAQ #29: What does DO-178C section 2.5.5.b (DO-278A section 2.6.5.b) mean, when it addresses requirements related to a default mode, if one is provided to protect against software load errors?**

Reference: DO-178C: Section 2.5.5.b
DO-278A: Section 2.6.5.b

Keywords: default mode; load

Answer:

DO-178C section 2.5.5.b (DO-278A section 2.6.5.b) states: *“If a system recovers to a default mode or safe state upon detection of a corrupted or inappropriate software load, then each partitioned component of the system should have safety-related requirements specified for recovery to and operation in this mode. Interfacing systems may also need to be reviewed for proper operation with the default mode.”*

When loading new software into a system, a number of problems can arise including interrupted loads and incorrect software loads. In most cases, the system designer will have considered these problems and arrived at a mechanism for ensuring a known system state can be established.

DO-178C Section 2.5.5.b (DO-278A section 2.6.5.b) is simply saying that the requirements for recovering to this known state have been captured. These requirements are, by definition, considered safety-related requirements since they establish the mechanism by which the system can be returned to a safe state and that the problem encountered during loading is contained. Note that interfacing systems may need to be reviewed for proper operation with the default mode.

- 3.30 FAQ #30: What does DO-178B section 2.6a(2) mean regarding system safety requirements addressing system anomalous behavior?**

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

- 3.31 FAQ #31: How does verification of product relate to “compiler acceptability”?**

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.32 FAQ #32: What are defensive programming practices?

Reference: DO-178C/DO-278A: Sections 1.4 and 4.5

Keywords: defensive programming; software development standards

Answer:

Note 1 of DO-178C/DO-278A section 4.5 mentions defensive programming when it states, “*Defensive Programming practices may be considered to improve robustness.*” Section 1.4.m of DO-178C/DO-278A states that notes are intended to “*provide explanatory material, emphasize a point, or draw attention to related items that are not entirely within context.*”

Defensive programming practices are techniques that may be used to prevent code from executing unintended or unpredictable operations by constraining the use of structures, constructs, and practices that have the potential to introduce failures in operation and errors into the code.

Defensive programming is related to the robustness of the software, not its correctness or maintainability. It should be acknowledged that “good” programming practice will promote robust software and hence could be considered defensive programming. Additionally, it should be noted that the robustness of the software should be defined by software requirements specifying the correct software response to abnormal conditions and inputs. Defensive programming may be considered in the design and coding standards.

Note: Reference the Object-Oriented Technology and Related Techniques supplement for related techniques that may provide supporting information.

Some examples of defensive programming are included below (this is not an exhaustive list):

- a. Avoidance of input data errors: To minimize input data errors:
 1. Check errors or bounds in data input whether it is from a human user or from another system.
 2. Use reasonableness checks on the incoming data.
 3. Account for potential data buffer overflow conditions (for example, queuing controls) and data aging (for example, time-stamps).
 4. Account for data type coercion, scaling, and loss of precision or accuracy that may occur in the interface between systems.
 5. Include run-time checks that allow detection of errors (for example, due to incorrect implementation).
- b. Avoidance of non-determinism: Non-determinism may be reduced through one or more of the following practices:
 1. Choose a language with a well-defined standard.

2. Do not permit self-modifying code.
 3. Minimize memory paging and swapping.
 4. Avoid use of dynamic binding.
 5. Do not assume initial values for variables that are not explicitly initialized.
 6. Avoid use of memory allocation and deallocation or when applicable, use the guidance of the Object Oriented Technologies supplement.
- c. Avoidance of complexity: Complexity may be reduced through the following practices:
1. Maximize cohesion and minimize coupling.
 2. Use overloading of operators and variable names cautiously.
 3. Use a single point of entry and exit for subprograms.
 4. Minimize use of interrupt-driven processing.
 5. Minimize use of multi-tasking and multi-processing in the architecture.
 6. Beware of side effects in the use of built-in functions and compiled libraries.
- d. Avoidance of interface errors: Interface errors may be minimized through the following practices:
1. Minimize the complexity of interfaces (for example, a large number of arguments, cryptic names, etc.).
 2. Use a standard set of units and precision throughout the software application.
 3. Avoid type coercion (for example, implicit or automated type conversions, mixed mode operations, fixed point scaling, etc.) in interfaces between procedures or follow the guidance of the Object Oriented Technologies supplement.
 4. Restrict use of global variables.
- e. Avoidance of logical errors: To minimize the potential for inducing these errors:
1. Understand possible loss of accuracy during computation.
 2. Be mindful of conversion errors (for example, fixed point scaling).
 3. Avoid using expressions with side effects such as assignment statements within evaluation statements.
 4. Verify the proper implementation of computed indices and array-bounds.
 5. Handle exceptions locally and uniformly.

3.33 FAQ #33: Is it permissible to NOT meet the safety objectives by justifying any deviations from the design standards?

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.34 FAQ #34: What is the concept of independence as used in DO-178B?

This FAQ has been removed, as approved by the SC-205/WG-71 committee. The information has been incorporated into DP #19.

3.35 FAQ #35: What are low-level requirements and how may they be tested?

Reference: DO-178C/DO-278A: Sections 5.0, 5.5, 6.0, 6.5, and Annex A

Keywords: architecture; derived requirements; high-level requirements; low-level requirements; requirements-based testing; testing; traceability

Answer:

The software requirements for a particular project are developed from system requirements, hardware architecture-dependent software requirements, and any additional derived requirements.

“During the software design process, the software architecture is defined and low-level requirements are developed” (reference DO-178C/DO-278A section 5.0). The aim of the architectural design is to break down the functions of the software requirements so that they can be regrouped into a logical software structure.

Once the architectural design is completed, the detailed design of the individual software functions is initiated. During this process, low-level requirements and any additional derived requirements are developed. These requirements are those from which the Source Code will be generated. *“However, if Source Code is generated directly from high-level requirements, then the high-level requirements are also considered low-level requirements...”* (reference DO-178C/DO-278A section 5.0). In order to enable the verification of complete implementation of the requirements, to give visibility to derived requirements, and to support the requirements-based test coverage analysis, Trace Data should be developed (reference DO-178C/DO-278A sections 5.5 and 6.5).

Compliance of the executable object code to the low-level requirements may be demonstrated using requirements-based tests. Test cases are developed for all levels of requirements. Then the requirements coverage analysis assesses the coverage of all requirements by the test cases.

3.36 FAQ #36: What is the definition or interpretation of derived requirements in DO-178C/DO-278A?

Reference: DO-178C/DO-278A: Section 5.0 and Annex B (glossary)

Keywords: derived requirements; system safety assessment process

Answer:

The DO-178C/DO-278A glossary defines derived requirements as: *“Requirements produced by the software development processes which (a) are not directly traceable to higher level requirements, and/or (b) specify behavior beyond that specified by the system requirements or the higher level software requirements.”*

During the software development process, requirements may be developed that do not trace directly to higher level requirements. Development of these derived requirements is typically based upon design, performance, or architectural decisions and therefore not directly traceable to higher level requirements. Even though a derived requirement may not be traced to higher level requirements, the derived requirement may affect the higher level requirements. DO-178C/DO-278A section 5 gives the following example: *“the need for interrupt handling software to be developed for the chosen target computer.”* In this example, the need for interrupt handling software will not appear in the higher level requirements, but it will be specified in the lower level requirements as derived requirements.

During the software development process, requirements may be developed that specify additional behavior that is not specified by the system requirements or higher level requirements. DO-178C/DO-278A section 5 gives the following example: *“The addition of scaling limits when using fixed point arithmetic.”* While the arithmetic to be performed would be specified in requirement(s), the use of fixed point arithmetic and the scaling limits on the fixed point arithmetic would be specified in a derived requirement if not specified in the higher level requirement(s). This derived requirement may be considered traceable to the higher level requirement(s) but it does add behavior that limits the values that may be used in the arithmetic operations. These limits need to be made *“available to the system processes including the system safety assessment process.”*

3.37 FAQ #37: What is meant by providing derived software requirements to the system processes, including the system safety assessment process?

Reference: DO-178C/DO-278A: Sections 2.2.1.g, 2.2.2.a, 2.2.3.a, 5.1.1.b, 5.1.2.i, 5.2.1.b, 5.2.2.c, and 11.6.d

Keywords: derived requirements; system safety assessment process

Answer:

All derived requirements are provided to the system processes, including the system safety assessment process, to ensure that the implementation of the derived requirements is visible to the system processes and that it does not adversely affect system safety (for example, produce a new hazard). The applicant should have a means for communicating the derived requirements to the system processes, including the system safety assessment process. The method that the applicant chooses to use for communicating the derived

requirements may be defined by the applicant's Software Requirements Standards, Software Design Standards, and/or Software Development Plan.

3.38 FAQ #38: What is the difference between Integration Process and Integration Testing?

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.39 FAQ #39: What is the definition of an unbounded recursive algorithm in DO-178C/DO-278A section 6.3.3.d?

Reference: DO-178C: Sections 6.3.3.d and 6.3.4.f and Annex B
DO-278A: Sections 6.3.3.d, 6.3.4.f, 12.4.11.2.4, and Annex B

Keywords: recursion; unbounded recursive algorithm

Answer:

An unbounded recursive algorithm is an algorithm that directly invokes itself (self-recursion) or indirectly invokes itself (mutual recursion) and does not have a mechanism to limit the number of times it can do this before completing (see DO-178C/DO-278A Annex B glossary).

The motivation for this FAQ is a consequence of the question: "Can recursive algorithms be used in airborne applications?" The answer to this question is "yes." However, if a recursive algorithm is used, the algorithm should be shown to be bounded (see DO-178C/DO-278A section 6.3.3.d). There are two reasons for this. The first is that the maximum execution time of the algorithm should be able to be determined, so that the execution of the algorithm can be shown to be within the real-time allocation for the function performed by the algorithm. Thus, a mechanism to establish an upper bound on the number of recursive calls is needed. Secondly, the stack space allocated to support recursion is limited by the actual stack space that is available. It should be shown that the stack space is available to handle the maximum level of recursion. Thus, the restriction of available stack space also requires that the recursive algorithm be bounded (see DO-178C/DO-278A section 6.3.4.f).

3.40 FAQ #40: What representation of the software is used to perform reviews and analyses of Source Code?

Reference: DO-178C/DO-278A: Section 6.3.4

Keywords: Source Code analysis; Source Code review

Answer:

When performing a review and analysis of the Source Code, the software should be reviewed in the representation as written in the programming language. It may be necessary to include other programming-related items (for example, linker directives, make files, etc.) in the review, as well as the other inputs to a software review in order to determine that the review objectives are satisfied. Source Code review and/or analysis is not concerned with examining the output of compilers and linkers/assemblers which are verified by other means.

3.41 FAQ #41: Why is Source Code to object code traceability required for Level A software?

This FAQ has been removed and is superseded by DP #12 (in section 4.12 of this document).

3.42 FAQ #42: What needs to be considered when performing structural coverage at the object code level?

Reference: DO-178C/DO-278A: Sections 4.5, 6.4.4.2, and 12.2

Keywords: object code; object code coverage; OCC; Source Code; structural coverage; tool qualification

Answer:

The main consideration is to demonstrate that the coverage analysis conducted at the object code level will provide the same level of confidence as that conducted at the Source Code level.

- a. To achieve this same level of confidence, the analysis needs to confirm the following, as a minimum:
 1. How the compiler produces the object code expected as assumed in the object code coverage (OCC) approach (for example, short-circuit behavior, compiler assumptions, compiler options and optimizations, etc.).
 2. How compiler options and optimizations impact the coverage achieved.
 3. How the results achieved from the OCC method provide the appropriate coverage assurance.
 4. How each of the following is addressed:
 - Control from outside to inside a function.
 - Jump statements (for example, break, continue, return, goto).
 - Bitwise operators, for example if Software Code Standards prohibit their use inside decisions.
 5. The need for qualification of tools used in OCC approach, for example, special test tools or in-circuit hardware devices.
 6. If necessary, specific design and code rules to enforce this same level of confidence are included in the Software Design Standards and Software Code Standards.
- b. Plans and standards need to support the analysis explained above and should address the following:

1. Grammar rules, coding restrictions and limitations, compiler restrictions and limitations, complexity limitations, etc. that are necessary to ensure that OCC will provide the appropriate structural coverage.
2. Any complexity and software architecture limitations for ensuring the validity of the OCC approach (for example, number of nested if statements, number of conditions in a decision, nested function calls, etc.).
3. Compliance with the rules, restrictions, and limitations needed for the OCC approach.

3.43

FAQ #43: What is the intent of structural coverage analysis?

Reference: DO-178C/DO-278A: Sections 6.4.4.2 and 6.4.4.3; Figure 6-1; and Annex A Table A-7

Keywords: intended function; structural coverage; verification

Answer:

DO-178C/DO-278A sections 6.4.4.2 and 6.4.4.3 define the structural coverage analysis activities and the possible resolution for code structure that was not exercised during requirements-based testing.

The purpose of structural coverage analysis, along with the associated structural coverage analysis resolution, is to complement requirements-based testing with the following:

1. Provide evidence that the code structure was verified to the degree required for the applicable software level.
2. Provide a means to support demonstration of absence of unintended functions.
3. Establish the thoroughness of requirements-based testing.

With respect to intended function, evidence that testing was rigorous and complete is provided by the combination of requirements-based testing (both normal range testing and robustness testing) and requirements-based test coverage analysis.

Requirements-based testing alone cannot completely verify the absence of unintended functionality. Code that is implemented without being linked to requirements may not be exercised by requirements-based tests. Such code could result in unintended functionality. Therefore, something additional needs to be done since unintended functions could affect safety. A technically feasible solution was found in structural coverage analysis.

The rationale is that if requirements-based testing proves that all intended functions are properly implemented, and if structural coverage analysis demonstrates that all existing code is reachable and adequately tested, these two together provide a greater level of confidence that there is no unintended functionality. Structural coverage analysis will:

- indicate to what extent the requirements-based test procedures exercise the code structure, and
- reveal code structure that was not exercised during testing.

Note: In the above text, the term “exercised during requirements-based testing” does not only mean that the specific code was exercised. It also means that the behavior of the code has been verified with respect to the requirements.

3.44 **FAQ #44: Why is structural testing not a DO-178C/DO-278A requirement?**

Reference: DO-178C/DO-278A: Sections 6.4.4.2 and 6.4.4.3

Keywords: structural coverage; structural testing

Answer:

There is a distinction between structural coverage analysis and structural testing. Whereas structural testing is the process of exercising software with test scenarios written from the Source Code, DO-178C/DO-278A section 6.4.4.2 explains that structural coverage analysis “determines which code structure, including interfaces between components, was not exercised by the requirements-based test procedures.” Structural testing does not satisfy the structural coverage analysis objectives.

The correct approach when structural coverage analysis identifies untested code is to consider the possible causes in accordance with DO-178C/DO-278A section 6.4.4.3, Structural Coverage Analysis Resolution. If any additional testing is required, it should be requirements-based testing, using high-level, low-level, or derived requirements, as appropriate.

Structural testing verifies that the Executable Object Code complies with the Source Code. Since the starting point for developing structural test cases is the code itself, there is no way of finding requirements (high-level, low-level, or derived) not implemented in the code through structural tests. It is a natural tendency to consider outputs of the actual code (which is de facto the reference for structural testing) as the expected results. This bias is avoided when expected outputs of a tested piece of code are determined by analysis of the requirements.

Since the code itself is used as the basis of the test cases, structural testing may fail to find simple coding errors.

3.45 **FAQ #45: What is the relevance of the exception case stated in the definition of dead code?**

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.46 FAQ #46: What is the meaning of section 7.2.2.g in DO-178C/DO-278A, when it states that a baseline or configuration item should be traceable either to the output it identifies or to the process with which it is associated?

Reference: DO-178C/DO-278A: Section 7.2.2.g

Keywords: baseline; configuration item; traceable

Answer:

The first step in understanding the meaning of this sentence is to realize that DO-178C/DO-278A glossary defines a baseline as one or more configuration items. A configuration item is one or more software components or software life cycle data treated as a unit for software configuration management purposes. DO-178C/DO-278A section 7.2.2.g states that a configuration item should be traceable to at least one of two items: (1) “*the output it identifies*”, or (2) “*the process with which it is associated*”. Both are described below.

The “*output it identifies*” is applicable for configuration items that are used as inputs. For example, the Software Requirements Data, the Software Development Plan, and the Software Design Standards are all inputs to the software design process. Therefore, these configuration items (the Software Requirements Data, Software Development Plan, and Software Design Standards) should be traceable to the “*output it identifies*,” namely, the Design Description.

The “*process with which it is associated*” is applicable for configuration items that are outputs. For example, the Design Description is the output of the software design process. The software design process uses the Software Requirements Data, the Software Development Plan, and software development standards as inputs. Therefore, the output configuration item, the Design Description, should be traceable to the “*process with which it is associated*,” the software design process, by its associated software life cycle data: the Software Requirements Data, the Software Development Plan, and the software development standards.

3.47 FAQ #47: What is meant by the term “certification credit” or “approval credit”?

Reference: DO-178C/DO-278A: Sections 8.3.a and Annex B

Keywords: approval credit; certification credit; evidence of compliance; means of compliance

Answer:

DO-178C and DO-278A use different terminology; therefore, each is discussed separately below.

DO-178C “certification credit”:

The DO-178C Annex B glossary defines “certification credit” as: “*Acceptance by the certification authority that a process, product, or demonstration satisfies a certification requirement.*”

For software, the applicant proposes their means of compliance for satisfying the software aspects of the certification requirements in the Plan for Software Aspects of Certification (PSAC). In other words, the applicant proposes processes, activities, methods, tools, etc. for which they are seeking “credit” for satisfying the software aspects of certification. Then, to achieve credit, the applicant performs the processes, satisfies the objectives, provides demonstrations, and produces the evidence of compliance, which are summarized in the Software Accomplishment Summary (SAS). Since certification credit is provided by “*acceptance by the certification authority*,” the certification authority’s approval of the SAS, design data, test data, etc. (that is, evidence of compliance) constitutes the granting of certification credit for the software.

Note: The applicant may conduct other processes and activities whose results are not submitted for certification credit.

DO-278A “approval credit”:

The DO-278A Annex B glossary defines “approval credit” as: “*Acceptance by the approval authority that a process, software product, or demonstration satisfies an applicable requirement.*”

For software, the applicant proposes their means of compliance for satisfying the software aspects of the approval requirements in the Plan for Software Aspects of Approval (PSAA). In other words, the applicant proposes processes, activities, methods, tools, etc. for which they are seeking “credit” for satisfying the software aspects of approval. Then, to achieve credit, the applicant performs the processes, satisfies the objectives, provides demonstrations, and produces the evidence of compliance, which are summarized in the Software Accomplishment Summary (SAS). Since approval credit is provided by “*acceptance by the approval authority*,” the approval authority’s approval of the SAS, design data, test data, etc. (that is, evidence of compliance) constitutes the granting of approval credit for the software.

Note: The applicant may conduct other processes and activities whose results are not submitted for approval credit.

3.48 FAQ #48: In addition to sections 9 and 10 of DO-178C, which provide a high-level overview of the certification liaison and certification processes, where can further guidance on the approval process for software be found?

Reference: DO-178C: Sections 9 and 10

Keywords: approval authority; approval process; certification authority; certification basis; certification liaison; certification processes; guidance

Answer:

Further guidance, information and appropriate links for issues related to the certification processes may be found by searching the web-sites of the Federal Aviation Administration (FAA) (<http://www.faa.gov>), the European Aviation Safety Agency (EASA) (<http://www.easa.europa.eu>), or the reader’s local certification or approval authority. Additional information (for example, policy and guidance materials such as Advisory Circulars, Orders, and Notices) may also be found through these web-sites. Additional information, such as the certification basis for a particular aircraft type, should be requested from the applicant’s local certification authority.

Note: Web-site addresses occasionally change. If the above web-sites do not connect, perform a web search or contact your local certification authority for an address update.

3.49 FAQ #49: Where can current certification authority guidance regarding issues not covered in DO-178B or expanding upon issues in DO-178B be found?

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.50 FAQ #50: What data items are deliverable to the certification/approval authority to support software approval and product certification/approval?

Reference: DO-178C/DO-278A: Sections 9.3 and 11

Keywords: approval authority; certification authority; data item; software life cycle data

Answer:

DO-178C/DO-278A section 9.3 specifies that the minimum set of software life cycle data submitted to the certification/approval authority includes the Plan for Software Aspects of Certification/Plan for Software Aspects of Approval, the Software Configuration Index, and the Software Accomplishment Summary. The certification/approval authority may, at their discretion, request additional software life cycle data items. Such a decision to request additional software life cycle data items is normally based upon the software level/assurance level of the system and the certification/approval authority's knowledge of the developing organization's life cycle processes.

Software data is only a part of the data needed to be submitted by the applicant for "product certification" or "product approval". Other data might include: system certification/approval plans, system validation and verification plans, flight test plans, safety analyses, system type design data, verification and validation results, flight test results, etc.

3.51 FAQ #51: What is meant by the term "type design," as used in section 9.4 of DO-178B?

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.52 FAQ #52: Why do the certification/approval authorities not approve an organization's process once, rather than approve each product submitted as part of a certification/approval application?

Reference: DO-178C/DO-278A: Section 10.3

Keywords: process approval; product

Answer:

The regulatory responsibility of the certification/approval authority is to approve the product. However, the certification/approval authority may, at their discretion, adjust the level of involvement in the review and approval process associated with a particular application. This adjustment is typically based on the certification/approval authority's previous experience with the applicant. Therefore, applicants lacking extensive

experience may be subjected to greater certification/approval authority involvement than more experienced applicants.

3.53 FAQ #53: Do the data items need to be prepared and packaged as specified in section 11 of DO-178B?

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.54 FAQ #54: Is the documentation required in DO-178C/DO-278A section 11 excessive, especially for small projects?

Reference: DO-178C/DO-278A: Section 11; Annex A

Keywords: documentation; small projects

Answer:

It is not the software's size but its impact on safety (and thereby the objectives to be met) that determine the type and quantity of data to be produced.

3.55 FAQ #55: What are the control category considerations when determining how to package the data items discussed in section 11 of DO-178C/DO-278A?

Reference: DO-178C/DO-278A: Section 7.3, 11, and Annex A

Keywords: assurance level; control category; CC1; CC2; software level

Answer:

The applicant is responsible for ensuring that the data items are controlled at the appropriate level. In some situations, it may be useful to combine data that have differing control categories into a single data item. When this is done, the data item should be controlled in accordance with the guidance for the higher control category.

A DO-178C example follows: An applicant decides that some Plan for Software Aspects of Certification (PSAC) content, as defined by DO-178C section 11, is to be included in the Software Development Plan (SDP). There is not an issue for Levels A and B systems since both documents are CC1, but for Levels C and D systems the SDP is controlled as CC2. This would mean the SDP should be controlled as CC1 to accommodate the control needed on the data that would otherwise be contained in the PSAC. Additionally, the SDP may need to be provided to the certification authority, as it contains information that is typically in the PSAC and may need certification authority approval.

A similar DO-278A example follows: An applicant decides that some Plan for Software Aspects of Approval (PSAA) content, as defined by DO-278A section 11, is to be included in the Software Development Plan (SDP). There is not an issue for AL 1 and 2 systems since both documents are CC1, but for AL 3, 4, and 5 systems the SDP is controlled as CC2. This would mean the SDP should be controlled as CC1 to accommodate the control needed on the data that would otherwise be contained in the PSAA. Additionally, the SDP may need to be provided to the approval authority, as it contains information that is typically in the PSAA and may need approval by the authority approval.

3.56 FAQ #56: How are redundancies inherent in the software verification documents eliminated?

Reference: DO-178C/DO-278A: Sections 11.b, 11.3, 11.13, and 11.14

Keywords: documentation; verification

Answer:

There are no intended redundancies in the data required by DO-178C/DO-278A.

The three types of verification data stated in DO-178C/DO-278A are the Software Verification Plan (see DO-178C/DO-278A section 11.3), the Software Verification Cases and Procedures (see DO-178C/DO-278A section 11.13), and the Software Verification Results (see DO-178C/DO-278A section 11.14). The terms documentation and documents are avoided in DO-178C/DO-278A, since the output of the verification process may take different forms as discussed in DO-178C/DO-278A section 11.b.

The output of the verification process is used to show that the verification process objectives have been met. The data can exist in a single data item and can be referenced instead of repeating information.

3.57 FAQ #57: Is it necessary to mention all the additional considerations in the Plan for Software Aspects of Certification (PSAC) or Plan for Software Aspects of Approval (PSAA) or is it sufficient to mention only the applicable additional considerations?

Reference: DO-178C/DO-278A: Sections 11.1.g and 12

Keywords: additional considerations; plans

Answer:

In the PSAC/PSAA, the applicant need only address the additional considerations found in DO-178C/DO-278A section 11.1.g and 12 that affect the certification/approval process. There is no specific guidance within DO-178C requesting the applicant to mention all the non-applicable additional considerations. However, the applicant may choose to list all the additional considerations in order to minimize questions from the certification/approval authority regarding their applicability.

It should also be noted that the list of additional considerations in DO-178C/DO-278A is not exhaustive. Other issues may arise, depending on the nature of the project, and should be identified in the PSAC/PSAA.

3.58 FAQ #58: How do you implement re-verification?

Reference: DO-178C/DO-278A: Sections 11.3.h and 12.1.1.e

Keywords: re-verification

Answer:

Re-verification is the process of determining what specific verification activities should be performed to ensure that a change to previously verified software was the correct

change and that the change did not adversely affect other software components or characteristics. Re-verification consists of these activities:

- a. An analysis is performed to determine the software components and characteristics that may be affected by the change.
- b. Appropriate verification items (for example, requirement and/or design reviews, test procedures and cases, timing analysis, memory analyses, etc.) are identified that should be performed to ensure the change was implemented correctly in the appropriate life cycle data, and that no adverse effects exist because of the change. This may include the development of new test procedures and cases.
- c. The identified verification is performed, the results recorded, and a determination made that all applicable verification objectives have been satisfied for the change.

3.59 FAQ #59: What type of non-flight software is covered by DO-178C?

Reference: DO-178C: Sections 1.2 and 12.2

Keywords: non-flight software; tool qualification

Answer:

The scope of DO-178C is limited to software aspects of airborne systems and equipment certification. Therefore, DO-178C objectives apply to the airborne software, related life cycle data, and tools used directly in the development and verification of the airborne software. DO-178C does not apply to, nor does regulatory guidance currently suggest the use of DO-178C on any other non-flight software, though there may be a safety issue from the use of the non-flight software.

Nevertheless, there are some cases where the applicant may choose to use DO-178C, even though it is not required by the regulatory authority. Also, some contracts might be issued that require DO-178C for non-flight software even though the regulations do not.

The only type of non-flight software addressed by DO-178C is software tools used by software life cycle processes in the development of airborne software. The DO-178C section on tool qualification (see section 12.2) can be used to assess if a tool needs to be qualified, and the applicable Tool Qualification Level. Then, DO-330, Software Tool Qualification Considerations, defines the tool qualification process and the needed tool qualification data.

3.60 FAQ #60: Is a complete set of plans required for modifications of a system?

Reference: DO-178C/DO-278A: Sections 9.3 and 12.1.1

Keywords: change; modification; plans; reuse

Answer:

The guidance of DO-178C/DO-278A specifies the need to submit plans and data to support the software aspects of certification/approval. Many applicants have questions about what plans and data are required for changes to a system which has previously been approved to DO-178C/DO-278A. Additionally, some applicants make frequent minor

changes and have a question concerning the best approach to support certification/approval. More specifically, many applicants want to know if all documents need to be resubmitted for every change or if some documents can be omitted.

DO-178C/DO-278A section 12.1.1 addresses software modification. The objectives of DO-178C/DO-278A should be satisfied even if the product is a modification or derivative of a system that complies with the objectives of DO-178C/DO-278A. DO-178C/DO-278A does not preclude the reuse of the plans or other data items used to show compliance with DO-178C/DO-278A.

DO-178C/DO-278A and other regulatory guidance material allow flexibility in the documents and data submitted for certification/approval compliance. If the applicant is careful in producing their software plans, these documents and data may be reusable for changes (regardless of whether they are major or minor changes, made frequently or infrequently).

When changes are small in their scope and impact to the system, the applicant may be able to gain agreement with the certification/approval authority that a simplified data package is appropriate. However, this simplified package should still support all the objectives of DO-178C/DO-278A. The simplified data package may achieve this goal by using references to existing data.

When changes are large in scope or impact on the system, it is likely that more supporting data and analysis may be needed. This additional information may result in submitting new or updated plans, if the former do not support the scope of the change to the system or software.

In summary, DO-178C/DO-278A objectives should be satisfied for all changes. However, the applicant can minimize the need to resubmit data and plans, if the applicant properly develops and coordinates the original plans to allow for expected changes to the system and its software.

3.61 FAQ #61: What constitutes a development tool and when should it be qualified?

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.62 FAQ #62: What are the requirements for flight test analysis software and ground-based test software?

Reference: DO-178C/DO-278A: Section 12.2

Keywords: test software; tool qualification

Answer:

For the purpose of this FAQ, flight test analysis software and ground-based test software are software tools used to monitor or analyze performance or compliance data of a system.

Flight test analysis software or ground-based test software can be used to automate some verification activities such as requirements-based testing. When used as a tool that eliminates, reduces, or automates a DO-178C/DO-278A verification activity without its output being verified, then the guidance for tool qualification for the flight test analysis

software or ground-based test software is applicable (see DO-178C/DO-278A section 12.2). When the flight test analysis software or ground-based test software is not used for a DO-178C/DO-278A verification activity, such as when used to satisfy a system regulatory requirement, then qualification of the tool is outside the scope of DO-178C/DO-278A.

3.63 FAQ #63: For exhaustive input testing, the applicant should provide an analysis which confirms the isolation of the inputs to the software. What does it mean to confirm the isolation?

Reference: DO-178C/DO-278A: Section 12.3.1

Keywords: exhaustive input testing; isolation

Answer:

“To confirm the isolation” means to demonstrate by means of analysis and/or test that the defined set of valid inputs to the software (as defined by compliance with DO-178C/DO-278A item 12.3.1.b) are the only inputs allowable to the software to produce the defined set of valid outputs of the software (also as defined for DO-178C/DO-278A item 12.3.1.a). In other words, the inputs and outputs are bounded. This isolation may be achieved and/or enforced by the subject software itself, by a software component outside the subject software, or by another system component.

If the inputs can be bounded and independently isolated, then exhaustive input testing can be justified by testing the appropriate combinations of inputs and input equivalent classes to satisfy the verification coverage for the subject software.

3.64 FAQ #64: Is it sufficient to use different linker or loader to produce dissimilar versions for avionics software?

Reference: DO-178C/DO-278A: Section 12.3.2

Keywords: dissimilar software; linker; loader

Answer:

No, using a different linker and/or loader alone would not be a suitable way of producing multiple, dissimilar versions of software. DO-178C/DO-278A states that multiple, dissimilar versions of software may be produced by combinations of the techniques described in the bulleted text of DO-178C/DO-278A section 12.3.2. These techniques in combination are expected to minimize the probability of common cause errors. The reason for bullet 6 is to remind the reader that using the same linker and/or loader may compromise the independence of the dissimilar systems.

3.65 FAQ #65: What is meant by “equivalent software verification process activity” in DO-178C/DO-278A sections 12.3.2.4 (Tool Qualification for Multiple-Version Dissimilar Software) and 12.3.2.5 (Multiple Simulators and Verification)?

Reference: DO-178C/DO-278A: Sections 12.2.1, 12.3.2, 12.3.2.4, and 12.3.2.5

Keywords: dissimilar software; multiple-version dissimilar software; simulators; software verification process; tool qualification; tools

Answer:

DO-178C/DO-278A sections 12.3.2.4 and 12.3.2.5 deal with the impact on tool qualification for tools used in the framework of multi-version dissimilar software.

DO-178C/DO-278A section 12.2.1 states: “*The purpose of the tool qualification process is to ensure that the tool provides confidence at least equivalent to that of the process(es) eliminated, reduced, or automated.*”

When dissimilar tools are used for multiple-version dissimilar software, it may be possible to modify the tool qualification process. It should be demonstrated that the use of such dissimilar tools (with the modified qualification process) provides the same confidence that the use of a single tool whose process is not modified. This confidence should be understood as noted in DO-178C/DO-278A section 12.3.2; that is, “*that the software verification process objectives are satisfied and that equivalent error detection is achieved for each software version.*” This is the meaning of “equivalent software verification process activity” in DO-178C/DO-278A. This confidence depends on the independence of the dissimilar tools, similarity of actions performed by both tools, and equivalency of the verification process activities that the dissimilar toolset is intended to eliminate, reduce, or automate. DO-178C/DO-278A sections 12.3.2.4 and 12.3.2.5 establish the minimum independence guidance for the dissimilar tools.

Multiple-version dissimilar software is described in DO-178C/DO-278A section 12.3, “Alternative Methods”. As such, all proposals for modifying the tool qualification guidance should have certification authority agreement.

3.66 FAQ #66: What is the difference between certification, approval, and qualification?

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.67 FAQ #67: What is analysis of data coupling and control coupling?

Reference: DO-178C/DO-278A: Sections 6.3.3, 6.3.4, and 6.4.4; Table A-7 of Annex A; and Annex B

Keywords: control coupling; data coupling; verification

Answer:

The DO-178C/DO-278A glossary (see Annex B) contains the following definitions for data coupling and control coupling:

- *“Data coupling - The dependence of a software component on data not exclusively under the control of that software component.”*
- *“Control coupling - The manner or degree by which one software component influences the execution of another software component.”*

The verification of data and control coupling involves a combination of the following:

- Reviews and analysis of software architecture, as stated in DO-178C/DO-278A section 6.3.3.b.
- Reviews and analysis of Source Code, as stated in DO-178C/DO-278A section 6.3.4.b.
- Requirements-based testing, confirmed by structural coverage analysis, as stated in DO-178C/DO-278A section 6.4.4.d.

The focus of this FAQ is on the objective stated in DO-178C/DO-278A section 6.4.4.d, which is:

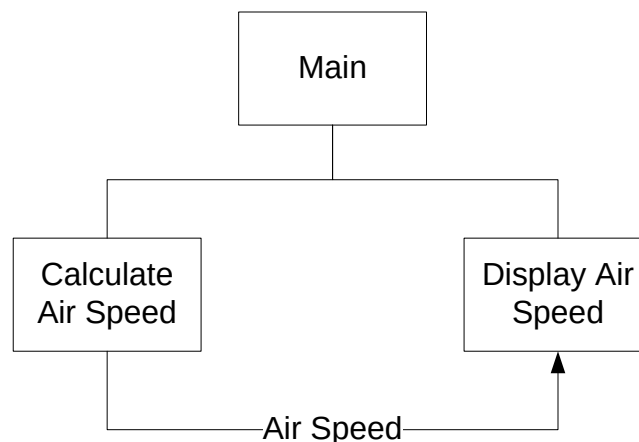
- *“Test coverage of software structure, both data coupling and control coupling, is achieved.”*

The related activity described in DO-178C/DO-278A section 6.4.4.2.c is:

- *“Analysis to confirm that the requirements-based testing has exercised the data and control coupling between code components.”*

The intent behind this objective is to ensure that applicants perform a sufficient amount of hardware/software integration testing and/or software integration testing.

The example below shows three components: a subprogram “Main”, which calls two subprograms “Calculate Air Speed” and “Display Air Speed”. The calculated air speed is passed between the two subprograms using a global variable “Air Speed”.



In this example, there exists control coupling between “Main” and “Calculate Air Speed”, and control coupling between “Main” and “Display Air Speed”. There also exists data coupling between “Calculate Air Speed” and “Display Air Speed”.

To satisfy the objective stated in DO-178C/DO-278A section 6.4.4.d, all of the following should be invoked during integration testing:

- The subprogram call from “Main” to “Calculate Air Speed”.
- The subprogram call from “Main” to “Display Air Speed”.
- The passing of "Air Speed" from "Calculate Air Speed" to "Display Air Speed".

These need not be three separate tests; they could be the same test.

3.68 **FAQ #68: The third sentence of the third paragraph of section 3.2 of DO-178C states that “*Component X illustrates the use of previously developed software used in a certified product.*” (The equivalent sentence in DO-278A states that “*Component X illustrates the use of previously developed software used in an approved system.*”) Is it necessary, for a reused component to have been used in the context of a previously certified product or an approved system?**

Reference: DO-178C/DO-278A: Sections 3.2 and 12.1

Keywords: component; previously developed software; PDS; reuse; software life cycle

Answer:

No. The intent of DO-178C/DO-278A section 3.2 is to illustrate a sequence of software development processes for several components of a single software product with different software life cycles. For further information on the reuse of software, refer to DO-178C/DO-278A section 12.1 and other FAQs or DPs on previously developed software (PDS).

3.69 **FAQ #69: What is the rationale to have software design process feedback to the planning process in section 5.2.2.g of DO-178C/DO-278A, where feedback to the system life cycle process and software requirements process seems adequate?**

Reference: DO-178C/DO-278A: Section 5.2.2.g

Keywords: plans; software design process; software planning process; software requirements process

Answer:

DO-178C/DO-278A section 5.2.2.g states: “*Inadequate or incorrect inputs detected during the software design process should be provided to the system life cycle process, the software requirements process, or the software planning process as feedback for clarification or correction.*” (Note: The underline and bold are added for emphasis.)

Whether or not the deficiencies are provided as feedback to the software planning process depends on the cause of the inadequate or incorrect input.

Software development and integral processes are defined during the software planning process. Inadequacies in either the development or integral processes can result in inadequate or incorrect inputs to the design process. Errors found in the design process such as those resulting from the misinterpretation of the software plans, incomplete or erroneous software standards, or incorrect or inadequate software requirements, may indicate the need for changes in the software planning process outputs themselves. That is, the planning process and/or the outputs of the planning process (the software plans and standards) may need to be modified to prevent similar deficiencies from continuing to occur.

3.70 FAQ #70: What is the purpose of the second sentence in DO-178C/DO-278A section 5.2.4.b?

Reference: DO-178C/DO-278A: Section 5.2.4.b

Keywords: deactivated code

Answer:

The second sentence of DO-178C/DO-278A section 5.2.4.b states: *"Unintended execution of deactivated code due to abnormal system conditions is the same as unintended execution of activated code."*

The purpose of the statement is to ensure that the same level of verification rigor is applied to in order to ensure the following:

- (1) That activated code is not inadvertently or erroneously activated by an abnormal system condition.
- (2) That deactivated code is not activated by an abnormal system condition.

Note: The definition of deactivated code is included in the DO-178C/DO-278A glossary (see Annex B).

3.71 FAQ #71: What is the purpose of traceability, how much is required, and how is it documented? For example, is a matrix required or are other methods acceptable?

This FAQ has been removed, as approved by the SC-205/WG-71 committee.

3.72 FAQ #72: What happens if an error indicates a weakness in the development process itself?

Reference: DO-178C/DO-278A: Sections 6.2.c, 7.2.3 through 7.2.5, and 11.17

Keywords: error; Problem Report; process weakness

Answer:

When the cause of an error is determined to be a process weakness, the defective process should be analyzed to determine whether it needs to be corrected and specific activities repeated.

The process weakness may indicate that similar errors occurred in other life cycle data produced by the same process. The impact of the errors and whether another process

would capture these errors will determine the extent and timing of the needed corrective action. The following questions should be addressed as part of this corrective action:

- Will another process detect the potential life cycle data errors? If so, the life cycle data previously verified would not need re-verification.
- Do the potential life cycle data errors affect the system operation or safety?
 - a) If not, recording a Problem Report and correcting the process for future work may be adequate.
 - b) If the errors do affect safety, system operation, or assurance level substantiation, an analysis should be performed to identify the other life cycle data produced with this process that may contain similar errors. These life cycle data may need to go through the corrective action process using a corrected process before seeking product approval.

3.73 **FAQ #73: Are timing measurements during testing sufficient or is a rigorous demonstration of worst-case timing necessary?**

Reference: DO-178C/DO-278A: Sections 6.3, 6.3.1.c, 6.3.2.c, 6.3.4.f, 6.4.2.2, 6.4.3.a, 11.9, 11.10, 11.20.i, 12.1.1; and Annex A Table A-5

Keywords: timing; worst-case execution time; WCET

Answer:

Background and Scope

Worst-case execution time (WCET) is the longest possible time that is needed to complete the execution of a set of tasks on a given processor in a real-time computational environment. Generally, the WCET addresses the total time used in a fixed time slice, and possibly over multiple time slices. Those time slices may be set by a real-time interrupt-driven clock, or some other deterministic manner.

There are factors which can complicate the calculation of WCET. Some of these include:

- Multiple interrupts that drive the system.
- Many decisions in the algorithms.
- Use of processor cache for instructions or data.
- Lack of a real-time driven clock interrupt.
- Differing methods of scheduling the tasks to be executed.
- Use of a real-time operating system.
- Operational behavior that is non-deterministic (for example stack usage).

These factors should be considered when the software architecture and requirements are defined to reduce or eliminate the complexities in analyzing the WCET.

In addition to verifying that the software requirements related to timing have been met, DO-178C/DO-278A Table A-5 objective 6 shows that the objective defined in section 6.3.4.f (which mentions worst-case timing) is applicable to Levels A, B, and C for DO-178C (AL1, 2, and 3 for DO-278A).

One cannot simply add the worst-case execution time from individual units and expect to obtain a meaningful system level assessment. The worst-case timing could be calculated by review and analysis of the Source Code and architecture, but impact of compiler (including options), linker (including options), and processor behavior should also be addressed. Timing measurements should be accompanied by an analysis demonstrating that the worst-case timing would be achieved; the analysis should consider processor behavior, such as cache performance.

WCET Assessment Methods

DO-178C/DO-278A discusses WCET and permits a variety of verification methods for determining the WCET for a system (including analyses and tests).

DO-178C/DO-278A Section 6.3, Software Reviews and Analysis, discusses the use of test on the software product to supplement reviews and analysis as a means to determine worst-case execution time.

DO-178C/DO-278A Section 6.3.4.f states that as part of meeting the verification objective of the Source Code being accurate and consistent, the worst-case execution timing should consider the impact of factors such as compiler options and some hardware features.

Related Timing Considerations

The Software Requirements Data and Design Description of DO-178C/DO-278A sections 11.9 and 11.10, respectively, require that timing requirements and constraints in 11.9.d and resource limitations in 11.10.e be determined. Although the term WCET is not used explicitly, WCET is certainly one aspect of these timing aspects to be addressed.

Similarly, DO-178C/DO-278A section 12.1.1.d identifies the need to analyze timing impact when modifying previously developed software.

WCET Results

DO-178C/DO-278A section 11.20.i identifies that the WCET data should be summarized and presented in the Software Accomplishment Summary.

3.74

FAQ #74: What is the difference between the development and life cycle objectives stated in DO-178C for Level A versus Level B software, and how does that relate to safety? Similarly, what is the difference in DO-278A for AL1 versus AL2 software?

Reference: DO-178C/DO-278A: Section 6.4.4.2 and Annex A

Keywords: AL1; AL2; Level A; Level B; independence; modified condition/decision coverage (MC/DC); Source Code to object code traceability; structural coverage