TOWARDS UNDERSTANDING THE DO-178C / ED-12C ASSURANCE CASE

C.M. Holloway

NASA Langley Research Center, Hampton VA, USA, c.michael.holloway@nasa.gov

Keywords: assurance case, software, standards, certification, confidence

Abstract

This paper describes initial work towards building an explicit assurance case for DO-178C / ED-12C. Two specific questions are explored: (1) What are some of the assumptions upon which the guidance in the document relies, and (2) What claims are made concerning test coverage analysis?

1 Introduction

For about two decades, compliance with *Software Considerations in Airborne Systems and Equipment Certification* (DO-178B / ED-12B) [7] has been the primary means for receiving regulatory approval for using software on commercial airplanes. Despite frequent and occasionally strident criticisms of the standard from various quarters, the empirical evidence is quite strong that it has been successful. Not only has no fatal commercial aircraft accident been attributed to a software error, many of the technological improvements that have been credited with significantly reducing the accident rate have relied heavily on software. For example, controlled flight into terrain—once one of the most common accident categories—has been nearly eliminated by Enhanced Ground Proximity Warning Systems, which are software-intensive [15].

The next edition of the standard, DO-178C / ED-12C, has been published by the issuing bodies [8]. New editions of two associated documents have also been published: *Software Integrity Assurance Considerations for Communication, Navigation, Surveillance and Air Traffic Management (CNS/ATM) Systems* (DO-278A / ED-109A) [10], and *Supporting Information* (DO-248C / ED-94C) [9]. Additionally four new guidance documents have been published to address software tool qualification considerations (DO-330 / ED-215) [11], model-based development and verification (DO-331 / ED-216) [12], object-oriented technology (DO-332 / ED-217) [13], and formal methods (DO-333 / ED-218) [14]. These standards have not yet received official regulatory authority approval, but the granting of approval is expected in due course.

The stated purpose of DO-178C / ED-12C remains essentially unchanged: providing guidance "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." In DO-178B / ED-12B little or no rationale is given for how a particular objective or collection of objectives contributes to achieving this purpose. Thus, the assurance case for the document is implicit. Empirical evidence suggests that this implicit assurance case is adequate, but its implicitness makes analysing why it is adequate quite difficult. DO-178C / ED-12C is also mostly rationale-free, but the revised edition of DO-248C / ED-94C includes a new section: 'Rationale for DO-178C [ED-12C] / DO-278A [ED-94C]'. This rationale section provides a basis from which building an explicit assurance case may be feasible.

This paper describes preliminary work towards building such an explicit assurance case for DO-178C / ED-12C. Two specific questions are explored: (1) What are some of the assumptions upon which the guidance in the document relies, and (2) What claims are made concerning test coverage analysis?

The remainder of the paper is organized as follows. Section 2 provides brief background material about the DO-178C / ED-12C document and the assurance case concept. Section 3 explores question (1). Section 4 discusses some initial possible answers to question (2). Section 5 explains potential future work and presents concluding remarks.

2 Background

The primary intended audience of this paper is people who are at least passingly familiar with both DO-178B / ED-12B and the assurance case concept. This section provides background information for readers who fall outside of this primary audience.

2.1 About DO-178C / ED-12C

Appendix A in DO-178C / ED-12C [8] contains a summary of the history of the DO-178 / ED-12 series of documents. The information below is derived from, and all quotations are taken from, this appendix.

The initial document in the series was published in 1982, with revision A following only three years later in 1985. Work on revision B began in the fall of 1989; the completed document, which was a complete rewrite of the guidance, was published in December 1992. This version introduced the notion of five

different possible software levels, with Level A denoting the highest level (on which the most rigorous objectives were levelled), and Level E denoting the lowest level (on which no objectives were levelled).

Twelve years after the adoption of DO-178B / ED-12B, RTCA and EUROCAE moved to update it, when they approved the creation of a joint special committee / working group in December 2004 (SC-205/WG-71).

This group began meeting in March 2005, and completed its work in November 2011. It operated under directions that called for (among other things) maintaining an "objective-based approach for software assurance" and the "technology independent nature" of the objectives. The special committee/working group was also directed to seek to maintain "backward compatibility with DO-178B / ED-12B" except where doing so would fail to "adequately address the current states of the art and practice in software development in support of system safety", "to address emerging trends", or "to allow change with technology." The documents produced by the efforts are listed above.

As a result of the terms of reference and operating instructions, DO-178C / ED-12C can be best thought of as an update to, as opposed to a re-write or substantial revision of, DO-178B / ED-12B. Differences between the documents include simple corrections of known errors and inconsistencies, changes in wording intended for clarification and consistency, an added emphasis on the importance of the full body of the document, a change in tool qualification criteria and the related creation of a separate document for tool qualification, modification of the discussion of system aspects related to software development, closing of some perceived gaps in guidance, and the creation of technology-specific supplements for formal methods, object-oriented technology, and model-based design and verification.

2.2 About the assurance case concept

The basic concept of an assurance case is simple[1]: provide a structured *argument* supported by *evidence* explaining why a particular *claim* about a system property is true. The most common instantiation of the concept involves claims about the system property of safety; hence the specific term *safety case* is perhaps more widely known than the more generic term.

Claims, arguments, and evidence constitute the three necessary components of an assurance case. Each of these components must be stated explicitly and clearly in order to produce a cogent assurance case. A critical aspect of an explicit and clear statement is articulating the context within and assumptions upon which the claims, arguments, and evidence depend.

Some existing approaches and notations for expressing assurance cases distinguish between context and assumptions [3]. For the purposes of this paper, we consider such a distinction to be unnecessary. Both refer to information that is not directly part of the explicit claims, arguments, or evidence, but without which the claims, arguments, and evidence cannot be understood fully or evaluated properly.

As a simple example of the importance of context and assumptions, consider the following claim: Improved helmet design will reduce the severity of concussions in football. Someone reading this claim in Edinburgh, Scotland, UK, is likely to find it unintelligible. "Helmets in football? There are no helmets in football!" In contrast, someone reading the same claim in Edinburgh, Indiana, USA, is likely to find it easy to understand. They will assume that the claim is to be interpreted within the context of American football, in which helmets are a required piece of equipment (aka kit).

Because of the importance of explicitly enumerating assumptions, one of the first activities that must be undertaken in trying to articulate the assurance case implicitly contained in DO-178C / ED-12C is to understand the context within and assumptions upon which the guidance rests. Initials steps towards this articulation are described in the next section.

3 Foundational assumptions

The work towards identifying all the relevant context and assumptions for the guidance has just begun. Thus far, four important categories have been discovered: the goal of satisfying airworthiness requirements; an implied relationship between safety and correctness; permission of process flexibility; and reliance on standard software engineering practices.

3.1 Satisfying airworthiness requirements

As noted in the introduction, the stated purpose of DO-178C / ED-12C is to "provide guidance 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" [8, p. 1] The document itself does not provide any additional details about what constitutes the airworthiness requirements. Users of the document are expected to know the specific requirements that apply to the system they are developing. These requirements must be included as a critical part of the context of any assurance case.

3.2 Relationship between safety and correctness

Section 2 of DO-178C / ED-12C and Section 5.2 of the Rationale make clear that the guidance is based on the assumption that adequate system safety processes have been followed in determining the requirements placed on the

[1] Although the concept is simple, much active research is on-going about how to best create, express, analyse, improve, and maintain assurance cases (for example, [1], [2], [4], [5], [19]).

software and its criticality level. For example, the Rationale states that "Software/assurance levels and allocated system requirements are a result of the system development and safety assessment processes" [9, p. 126]

These sections also make clear that all relevant safety-specific requirements are expected to be included. That is, one of the inputs that must be available before the guidance is applied is a comprehensive set of the requirements, including all of the requirements that must be satisfied to ensure an adequate level of safety is maintained. DO-178C / ED-12C is not concerned with determining or analysing these safety requirements, but only in satisfying them. Hence, it is strictly true, as is often asserted, that the standard is not a safety standard [6]. Conducting system safety analysis is intentionally outside of the scope of the guidance. Guidance for it is expected from other documents (for example [16], [17]).

A reader may thus ask how safety can be legitimately mentioned as an important part of the purpose of the guidance. The answer to this question is based on the following reasoning, which is not explicitly stated, but definitely implied. Given a set of requirements that includes everything necessary to provide an adequate level of safety, then ensuring that the requirements are met necessarily ensures that the adequate level of safety is provided. So, the guidance needs to be concerned only with ensuring that software satisfies its requirements. Within the context to which the guidance applies, software system correctness necessarily implies software system safety. This implication does not hold in the general case, but it does hold in this specific case. Thus, the DO-178C / ED-12C assurance case can concentrate on demonstrating correctness of implementation.

3.3 Permission of process flexibility

Another foundational assumption of DO-178C / ED-12C may come as a surprise to people whose only exposure to the guidance and its ancestors comes through criticisms by academics: developers are permitted wide process flexibility. As stated in the Rationale, "The committee wanted to avoid prescribing any specific development methodology. [The guidance] allows for a software life cycle to be defined with any suitable life cycle model(s) to be chosen for software development. This is further supported by the introduction of 'transition criteria'. Specific transition criteria between one process and the next are not prescribed, rather [the guidance] states that transition criteria should be defined and adhered to throughout the development life cycle(s) selected" [9, p. 126].

The DO-178C / ED-12C guidance does include detailed descriptions of specific activities that may be performed in order to satisfy particular objectives. However, the guidance also explicitly states that the activities themselves may be changed: "The applicant should plan a set of activities that satisfy the objectives. This document describes activities for achieving those objectives. The applicant may plan and, subject to approval of the certification authority, adopt alternative activities to those described in this document. The applicant may also plan and conduct additional activities that are determined to be necessary" [8, p. 3].

This flexibility must be considered in the creation of an assurance case. It means that certain parts of the argument should permit alternate instantiations. An instantiation based on the activities described in the guidance can be developed, but it should be made clear that this is only an example, and that other instantiations may be possible.

3.4 Reliance on standard software engineering practices

The fourth foundational assumption of DO-178C / ED-12C that has been uncovered thus far is that it relies in substantial part on the efficacy of standard software engineering practices. The overview section of the Rationale identifies this reliance clearly: "Since DO-178C / DO-278A heavily borrows from standard software engineering principles that are well understood, rationale is only provided for those elements within the document that are specific to aircraft certification (or CNS/ATM system approval). The reader is directed to the public literature for rationale for items not covered in this section" [9, p. 125].

In creating an assurance case, a decision must be made about how to handle those parts of the guidance for which the rationale lies in standard practice. One option is to terminate the analysis of such parts with a reference to practice. Another option is to continue the analysis by including claims, arguments, and evidence provided in the 'public literature' mentioned in the Rationale (such as [6] [18]).

4 Test Coverage Analysis

Besides exploring the assumptions underlying the DO-178B / ED-12C guidance, the other preliminary work that has been conducted thus far is considering a specific aspect of the guidance, namely test coverage analysis. This area was chosen because test coverage has been among the most frequently criticised aspects of DO-178B / ED-12B, and is likely to continue to be so for the updated guidance.

The particular question that guided the initial work was, "What claims are made concerning test coverage analysis?" A careful articulation of the actual claims concerning test coverage should help clarify whether the criticisms are valid, or simply based on misunderstandings. Valid criticisms will definitely affect the assurance case that is eventually produced, by identifying parts of the case in which confidence should not be placed. The potential effect on the assurance case of existing misunderstandings is less clear-cut.

Guidance for testing is provided in Section 6.4 [8, pp. 44-51], with test coverage analysis guidance given in Section 6.4.4 [8, pp. 49-51]. Testing objectives are summarised in Table A-6 [8, p. 101]; test coverage objectives are summarised in Table A-7 [8, p. 102]. *Supporting Information* [9] contains a

discussion in the Rationale section [9, p. 129-130] and several frequently asked questions and discussion papers related to test coverage:

- FAQ #42 What needs to be considered when performing structural coverage at the object code level? [9, p. 22]
- FAQ #43 What is the intent of structural coverage analysis [9, pp. 23 – 24]
- FAQ #44 Why is structural testing not a DO-178C / DO-278A requirement? [9, p. 24]
- 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? [9, pp. 38-39]
- DP #8 Structural Coverage and Safety Objectives [9, pp. 70 – 71].
- DP #13 Discussion of Statement Coverage, Decision Coverage, and Modified Condition/Decision Coverage (MC/DC) [9, pp. 81- 88].

The guidance and supporting information distinguishes between the purposes of testing and the purposes of test coverage analysis. Testing is intended "to demonstrate that the software satisfies its requirements and demonstrate that errors that could lead to unacceptable failure conditions, as determined by the system safety assessment process, have been removed" [8, p. 44]. The objectives associated with testing involve the relationship between executable object code and its requirements, along with the compatibility of the executable object code with the target computer. Testing is all about the software product itself.

Test coverage analysis, on the other hand, has different purposes. Two types of coverage analysis are described in the guidance: requirements-based test coverage analysis, and structural coverage analysis. The purpose of the former is simply to analyse the test cases that were used in the requirements-based testing to confirm that they satisfy the criteria of the guidance. The purpose of the latter is a bit less well understood. Hence the abundance of popular criticism of the structural coverage criteria, and the amount of space devoted to it in *Supporting Information*. Determining the structural coverage claims that should be included in an assurance case is difficult. The discussion in the rest of this section is only a beginning towards that determination.

Concerning structural coverage analysis, the guidance states that it "determines which code structure, including interfaces between components, was not exercised by the requirements-based test procedures. The requirements-based test cases may not have completely exercised the code structure, including interfaces, so structural coverage analysis is performed and additional verification produced to provide structural coverage" [8, p. 49].

It is important to recognize that structural coverage analysis is *not* presented in the guidance as a form of testing. It is presented as a means of determining whether the requirements-based tests covered the code to the extent required by the software level. If the analysis shows that adequate coverage has been achieved, no additional tests are required[2].

Evaluating the thoroughness of requirements-based testing is the purpose explicitly mentioned in the guidance. FAQ #43 mentions two additional purposes: providing "evidence that the code structure was verified to the degree required for the applicable software level", and providing "a means to support demonstration of absence of unintended functions."

Concerning the first of these additional purposes, the guidance requires demonstrating increasingly higher degrees of coverage for higher software level. Level D does not require any structural coverage analysis. Level C requires achieving statement coverage (every statement in the program is invoked at least once). Level B requires decision coverage (every entry and exit point to the program is invoked at least once and every decision in the program has taken on all possible outcomes at least once). For Level A software, achieving modified condition / decision coverage (MC/DC) is required (decision coverage with the additional requirement that "each condition in a decision has been shown to independently affect a decision's outcome" [8, p. 114]).

Intuitively, the notion of basing the thoroughness of coverage requirements on the criticality of the software makes sense. Executing more code structure should justify higher confidence that errors have not been missed than executing less. For the Level C and B requirements, the Rationale section [9, p. 130] provides little additional insight beyond this intuitive notion. For the Level C requirement it simply states that statement coverage was "deemed satisfactory", and for Level B it says that decision coverage "was considered sufficient to address the increase in the associated hazard category."

The Rationale's discussion about the reasons behind the MC/DC requirement does provide insight. MC/DC was introduced in DO-178B / ED-12B. Its introduction is identified as a compromise "based on experience gained from three aircraft programs, where an approach derived from hardware logic testing that concentrated on showing that each term in a Boolean expression can be shown to affect the result, was applied to software." This compromise was between the committee's desire that for level A software all logic expressions should be fully explored, and the recognition that "the use of techniques such as multiple condition decision coverage, or exhaustive truth table evaluation to fully explore all of the logic was … impractical."

[2] If someone says, for example, "You have to do MC/DC testing on Level A software," they are either using the language very loosely, or they do not know what they are talking about (or perhaps both). Anyone doubting the truth of this statement should consult FAQ #44 [9, p. 24].

Concerning demonstrating unintended function, structural coverage analysis serves to help close a gap that might be left by requirements-based testing. As FAQ #43 states, "Code that is implemented without being linked to requirements may not be exercised by requirements-based tests. Such code could result in unintended functionality" [9, p. 23]. Because unintended functions could conceivably have a negative impact on system safety, detecting and eliminating them increases in importance with higher software levels. Structural coverage analysis is intended as a means to increase confidence that the code that really exists in the software has been reached, and thus any unintended functionality has been exposed.

As noted at the beginning of this section, the motivating question for the initial exploration was "What claims are made concerning test coverage analysis?" Claims identified thus far include the following:

- Requirements-based test coverage analysis confirms that the requirement-based tests satisfy the criteria of the guidance.
- Structural coverage analysis confirms whether the requirements-based tests covered the code to the extent required by the software level.
- Structural coverage analysis identifies unintended functions that exist in the software.

Refinements and additions to these claims are likely to be made as the effort continues.

5 Future Work

This paper has described preliminary work towards building an explicit assurance case for DO-178C / ED-12C. The next steps to be followed include receiving feedback from readers of the paper; articulating the top-level claim of the assurance case; completing the determination of the assumptions underlying this claim, and deciding how to handle each of these assumptions in the assurance case; deciding what notation(s) to use; completing the test coverage analysis work; and determining whether to take a breadth-first or depth-first approach to discovering sub-claims, arguments, and evidence.

Once these steps are taken, the creation of a full assurance case can commence. Readers interested in collaborating in the endeavour are encouraged to contact the author.

References

[1] R. Bloomfield, P. Bishop. "Safety and Assurance Cases: Past, Present and Possible Future", *Making Systems Safer*, C. Dale and T. Anderson (eds), Springer-Verlag, pp. 51-67, (2010).

[2] P. Graydon, I. Habli, R. Hawkins, T. Kelly, and J. Knight. "Arguing Conformance", IEEE Software, **29** (3), pp. 50-57, (2012).

[3] GSN Community. GSN Community Standard Version 1, (2011). [http://www.goalstructuringnotation.info/documents/GSN_Standard.pdf] Visited 20 July 2012.

[4] R. Hawkins, T. Kelly, J. Knight, and P. Graydon. "A New Approach to Creating Clear Safety Arguments", *Advances in Systems Safety*, C. Dale and T. Anderson (eds), Springer-Verlag, pp. 3-23, (2011).

[5] C. M. Holloway. "Safety Case Notations: Alternatives for the Non-Graphically Inclined?" *Proceedings of the 3rd IET International System Safety Conference*, (2008).

[6] J. Knight. *Fundamentals of Dependable Computing for Software Engineers*. CRC Press, (2012).

[7] RTCA / EUROCAE. "Software Considerations in Airborne Systems and Equipment Certification", DO-178B/ED-12B (1992).

[8] RTCA / EUROCAE. "Software Considerations in Airborne Systems and Equipment Certification", DO-178C/ED-12C, (2011).

[9] RTCA / EUROCAE. "Supporting Information for DO-178C [ED-12C] and DO-178A [ED-109A]", DO-248C/ED-94C, (2011).

[10] RTCA / EUROCAE. "Software Integrity Assurance Considerations for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems", DO-278A/ED-109A, (2011).

[11] RTCA / EUROCAE. "Software Tool Qualification Considerations", DO-330/ED-215, (2011).

[12] RTCA / EUROCAE. "Model-Based Development and Verification Supplement to DO-178C [ED-12C] and DO-178A [ED-109A]", DO-331/ED-216, (2011).

[13] RTCA / EUROCAE. "Object-Oriented Technology and Related Techniques Supplement to DO-178C [ED-12C] and DO-178A [ED-109A]", DO-332/ED-217, (2011).

[14] RTCA / EUROCAE. "Formal Methods Supplement to DO-178C [ED-12C] and DO-178A [ED-109A]", DO-333/ED-218, (2011).

[15] J. Rushby. "New Challenges in Certification of Aircraft Software", *Proceedings of the 11th International Conference on Embedded Software (EMSOFT)*, pp. 211-218, (2011).

[16] Society of Automotive Engineers. *Guidelines for Development of Civil Aircraft and Systems*, SAE ARP 4754a, (2010).

[17] Society of Automotive Engineers. *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipement*, SAE ARP 4761, (1996).

[18] I. Sommerville. *Software Engineering*. 9th edition. Addison-Wesley, (2011).

[19] T. Yuan, T. Kelly. "Argument Schemes in Computer System Safety Engineering", *Informal Logic*, **31** (2), pp. 89-109, (2011).