Explicate '78: Uncovering the Implicit Assurance Case in DO–178C

C. Michael Holloway

NASA Langley Research Center

Hampton, VA, USA

Abstract *For about two decades, compliance with Software Considerations in Airborne Systems and Equipment Certification (DO–178B/ED–12B) has been the primary means for receiving regulatory approval for using software on commercial airplanes. A new edition of the standard, DO–178C/ED–12C, was published in December 2011, and recognized by regulatory bodies in 2013. The purpose remains unchanged: 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.' The text of the guidance does not directly explain how its collection of objectives contributes to achieving this purpose; thus, the assurance case for the document is implicit. This paper presents an explicit assurance case developed as part of research jointly sponsored by the Federal Aviation Administration and the National Aeronautics and Space Administration.*

1 Introduction

Software Considerations in Airborne Systems and Equipment Certification (DO–178B) (RTCA 1992)[1] was published in 1992. Compliance with this document has been the primary means for receiving regulatory approval for using software on commercial airplanes ever since. Despite criticisms of the DO–178B from various quarters, the empirical evidence suggests strongly that it has been successful, or at worst, has not prevented successful deployment of software systems on aircraft. Not only has no fatal commercial aircraft accident been attributed to a software failure, 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 software-intensive Enhanced Ground Proximity Warning Systems (Rushby 2011).

A new edition of the standard, DO–178C, was published by the issuing bodies in late 2011 (RTCA 2011a). New editions of two existing associated documents and four entirely new guidance documents were also published at the same time. More information about these documents is provided later in this paper. The relevant documents received official regulatory authority recognition in 2013 (Federal Aviation Administration 2013b, European Aviation Safety Agency 2013).

The stated purpose of DO–178C remains essentially unchanged from its predecessor: 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.' The text of the guidance provides little or no rationale for how it achieves this purpose. A new section in the revised edition of DO–248C (RTCA 2011b), 'Rationale for DO–178C / DO–278A', contains brief discussions of the reasons behind some specific objectives and collection of objectives; nevertheless, the overall assurance case for why DO–178C achieves its purpose is almost entirely implicit.

Although empirical evidence suggests that this implicit assurance case has been adequate so far, its implicitness makes determining the reasons for this adequacy quite difficult. Without knowing the reasons for past success, accurately predicting whether this success will continue into the future is problematic, particularly as the complexity and autonomy of software systems increases. Equally problematic is deciding whether proposed alternate approaches to DO–178C are likely to provide an equivalent level of confidence in safety.

As a potential way forward for addressing these problems, the Federal Aviation Administration (FAA) and the National Aeronautics and Space Administration (NASA) are jointly sponsoring an effort, called the Explicate '78 project within NASA, to uncover and articulate explicitly (that is, explicate) DO–178C's implicit assurance case. Early work in this effort was described in (Holloway 2012, Holloway 2013).

[1] The European Organisation for Civil Aviation Equipment (EUROCAE) uses a different document numbering scheme, but the content of the documents is equivalent. For example, DO–178C is equivalent to ED–12C. For simplicity, only the DO numbering scheme is used in the body of this paper. Also, please note that although once upon a time RTCA was an abbreviation for Radio Technical Commission for Aeronautics, since 1991 the four letters have been the freestanding name of the organization.

This paper describes the current status of the research, and is organized as follows. Section 2 provides background material. Section 3 presents the key concepts underlying, and several excerpts from, the explicit assurance case developed to date. Section 4 discusses the next steps in the research and makes concluding remarks.

2 Background

Fully understanding this paper requires at least a passing familiarity with DO–178B/C and the assurance case concept. This section provides background information on these two subjects for readers who do not already possess the requisite knowledge. This section also provides a brief discussion of prior related published work.

Although some excerpts from the assurance case are expressed using the Goal Structuring Notation (GSN), background material about GSN is not provided because of space limitations. Readers unfamiliar with GSN should consult (GSN Committee 2011).

2.1 About DO–178C

The information in this section is based on Appendix A in DO–178C, which contains a summary of the history of the DO–178 series of documents. The initial document in the 178 series was published in 1982, with revision A following in 1985. Work on revision B began in the fall of 1989; the completed document, which was a complete rewrite of the guidance from revision A, was published in December 1992. Among many other changes, the B version expanded the number of different software levels based on the worst possible effect that anomalous software behaviour could have on an aircraft. Level A denoted the highest level of criticality (for which satisfying the most rigorous objectives was required), and Level E denoted the lowest level (which was objective-free). The B version also introduced annex tables to summarize the required objectives by software level.

Twelve years after the adoption of DO–178B, RTCA and EUROCAE moved to update the document by approving the creation of a joint special committee / working group in December 2004 (SC-205/WG-71). This group started meeting in March 2005, and completed its work in November 2011. The terms of reference for the group called for (among other things) maintaining the 'objective-based approach for software assurance' and the 'technology independent nature' of the objectives. SC-205/WG-71 was also directed to seek to maintain 'backward compatibility with DO–178B' 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.'

Ultimately the effort produced seven documents. In addition to DO–178C, new editions were written of two existing, associated documents: DO–278A: Software Integrity Assurance Considerations for Communication, Navigation, Surveillance and Air Traffic Management (CNS/ATM) Systems (RTCA 2011c), and DO–248C: Supporting Information for DO–178C and DO–278A (RTCA 2011b). The former is very similar to DO–178C, but addresses software in certain ground-based systems, which operate within a different regulatory scheme from airborne systems. The latter provides answers to various questions and concerns raised over the years by both industry and regulatory authorities. It contains 84 frequently asked questions, 21 discussion papers, and, as noted above, a brief rationale.

Four new guidance documents were also published to address specific issues and techniques: DO–330: Software Tool Qualification Considerations (RTCA 2011d); DO–331: Model-Based Development and Verification Supplement to DO–178C and DO–278A (RTCA 2011e); DO–332: Object-Oriented Technology and Related Techniques Supplement to DO–178C and DO–278A (RTCA 2011f); and DO–333: Formal Methods Supplement to DO–178C and DO–278A (RTCA 2011g). The subject matter of these documents is evident from their titles.

As a result of the terms of reference and operating instructions under which DO–178C was developed, the document is only an update to, as opposed to a re-write or substantial revision of, DO–178B. Differences between the B and C versions include 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 qualification criteria for tools 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 the technology-specific supplements enumerated above for formal methods, object-oriented technology, and model-based design and verification.

2.2 About assurance cases

The concept of an assurance case is a generalization of the safety case concept. A common definition of a safety case is 'a structured argument, supported by a body of evidence that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given operating environment' (UK Ministry of Defence 2007). Claims are made concerning the achievement of an acceptable level of safety, and arguments and evidence are focused on providing justified confidence that those safety claims are satisfied. A more general assurance case is concerned about providing justified confidence that claims are satisfied about other desired attributes such as correctness, functionality, performance, or security.

Claims, arguments, evidence[2], context, and assumptions constitute five components of a well-structured assurance case (Knight 2012). *Claims* are statements about desired attributes. Other names that are used for the same concept include *goals*, *propositions*, and *conclusions*. In a full assurance case, there will likely be many claims that must be shown to hold, at varying levels of generality. An example of a high-level claim is The software performs its intended function at an acceptable level of safety (the Arial font is used throughout the paper to denote assurance case text). Examples of claims with an increasing level of specificity are as follows: High-level requirements are a satisfactory refinement of system requirements; Adequate configuration management is in place; and Configuration items are identified.

An *argument* explains how a stated claim is supported by, or justifiably inferred from, the evidence and associated subclaims. Other terms sometimes used for the same concept include *strategies*, *warrants* (Toulmin 2003), and *reasons*. Just as a system nearly always consists of multiple sub-systems, an argument nearly always consists of multiple sub-arguments; but the term sub-argument is almost never used.

Evidence refers to the available body of known facts related to system properties or the system development processes. *Data*, *facts*, and *solutions* are synonymous terms. Examples of evidence include hazard logs, testing results, and mathematical theorems.

Context generally refers to any information that is needed to provide definitions or descriptions of terms, or to constrain the applicability of the assurance case to a particular environment or set of conditions. As example, the context for the claim The software performs its intended function with a level of confidence in safety that complies with airworthiness requirements would likely include the applicable airworthiness requirements (Federal Aviation Administration 2013a), a description of the intended function of the software, and any constraints on the environment in which the software is expected to be used. Some recent research defines context more strictly than has been done previously (Graydon 2014).

Assumptions are statements on which the claims and arguments rely, but which are not elaborated or shown to be true in the assurance case. As an example, an argument concerning safety that shows that all identified hazards have been eliminated may rely on the assumption All credible hazards have been identified.

Claims, arguments, evidence, context, and assumptions are all present implicitly in the collective minds of the developers of any successful engineered system. An assurance case simply provides a means for ensuring that this implicit knowledge is documented explicitly in a form that can be examined carefully and critically, not only by the developers, but also by others. An active research community is exploring how to best create, express, analyze, improve, and maintain assurance cases. Examples include (Matsuno 2014, Ayoub et al. 2013, Denney et al. 2013, Hawkins et al. 2013, Rushby 2013, Goodenough et al. 2012, Yuan and Kelly 2011, Bloomfield and Bishop 2010, Hawkins and Kelly 2009, Holloway 2008).

2.3 Previous work

No published work was found that has attempted to accomplish the same goals as the current effort, but two previous projects did address related aspects concerning DO–178B and assurance cases.

The MITRE Corporation tried to map three different standards into an assurance case framework (Ankrum and Kromholz 2005). The primary purpose of this effort was to explore two primary hypotheses: all assurance cases have similar components, and an assurance standard implies the structure. One of the three standards used in the study was DO–178B. The created assurance case was structured rigidly around the DO–178B chapters. For example, the top-level claim was DO–178B Software Considerations are taken into account. Sub-claims were given for each of the DO–178B chapters 2 – 9; for example: 2.0 System Aspects are taken into account; 5.0 Software Development Process is executed as planned; and 9.0 Certification Liaison process is properly established & executed.

[2] The claims, argument, evidence distinction (perhaps using slightly different words) is well established within the literature. A strong case can be made that *argument* is more properly thought of as a broad term, of which *claims* and *evidence* are components; however, this particular paper is not the place to try to clean up the terminology, so the standard terms and distinctions are maintained.

The effort appears to have concentrated on translating the textual and tabular form of DO–178B into a graphical form with as little interpretation or abstraction as possible. This differs substantially from the current research, which is concentrating on discovering the underlying implicit assurance case, not rigidly translating one form of concrete expression into another form.

Researchers at the University of York and QinetiQ in the United Kingdom conducted the other related previous work (Galloway et al. 2005). The primary goal of this research was to explore ways to justify substitution of one technology for another. In particular, a major emphasis was placed on developing arguments showing that the evidence produced by replacements for testing (such as formal proof) could be at least as convincing as the evidence produced by testing. As part of this research, certain aspects of the testing-related objectives of DO–178B were explored and GSN representations were produced. Unpublished results from the research were submitted to SC–205/WG–71, and considered by the Formal Methods sub-group, which wrote the document that eventually become DO–333. This material was also considered during the process of developing the assurance case for DO–178C that will be discussed in the next section.

3 The explicit case

The first version of a complete, explicit assurance case in the Explicate '78 project was completed and expressed in GSN at the end of 2013. It was structured in a modular fashion, with separate arguments for each of the four main software levels A-D. To the extent consistent with the 178C text, arguments from lower software levels were referenced directly in the arguments for higher software levels. This version was reviewed in varying levels of detail and rigor by a handful of FAA personnel and other interested parties over a period of six months.

Revisions based on the review yielded a version (called e78-1.5) that was substantially similar in overall structure to the original, but which differed in some subtle ways and in several specific details. This version also introduced generic primary and confidence arguments, which were not strictly necessary, but which served to illustrate a consistent argument structure across levels. A lengthy presentation describing e78-1.5 was delivered to over 100 people at the FAA-sponsored 2014 National Systems, Software, and Airborne Electronic Hardware Conference in September 2014. Comments received at the conference prompted several minor modifications to the GSN structures, and the creation of textual representations of portions of the case, yielding version e78-1.6, which is the version described here.

The section is organized as follows. First, four fundamental concepts that greatly influence the structure and content of the e78-1.6 assurance case are discussed. Second, salient characteristics about the case itself are provided. Third, five excerpts from the case are presented.

3.1 Fundamental concepts

The following four concepts provide the foundation on which the explicit assurance case is built: transforming safety into correctness, allowing life cycle flexibility, using confidence arguments, and explicating before evaluating. The first two of these concepts permeate the DO–178C guidance itself. The latter two concepts arose as solutions to difficulties encountered in the early days of trying to structure an explicit assurance case. All four are discussed below.

3.1.1 Transforming safety into correctness

A fundamental assumption of DO–178C is discernable only through inferences from the text; it involves the relationship between safety and correctness. Although in the general case, these two concepts are not equivalent (Knight 2012), DO–178C rests implicitly on the assumption that within the constraints established by the guidance, establishing justifiable confidence in the correctness of the software with respect to its requirements *is* sufficient to establish justifiable confidence that the software does not contribute to unsafe conditions.

The validity of this assumption rests on the further assumption that adequate system safety processes have been followed in determining the requirements placed on the software and its associated criticality level. As stated in the Rationale: 'Software/assurance levels and allocated system requirements are a result of the system development and safety assessment processes' (RTCA 2011b, p. 9).

The system requirements allocated to software are further assumed by DO–178C to include all of the requirements that must be satisfied by the software to ensure an adequate level of safety is maintained. DO–178C 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. Conducting system safety analysis is intentionally outside the scope of the guidance. Guidance for it is expected from other documents (SAE International 1996, SAE International 2010).

Any new requirements that arise during software development must be passed back to the system processes, including system safety processes, for analysis of (among other things) potential safety implications. Such requirements were called *derived requirements* in DO–178B; the term is retained in 178C. (This choice of terminology has been a frequent source of confusion, because the phrase *derived requirements* is not commonly used in the broader software engineering community. When encountering the term for the first time, many people assume that it means requirements derived from higher level requirements, as opposed to new requirements that are explicitly *not derived* from higher level ones. Some members of SC–205/WG–71 tried, but failed, to change the terminology.)

With these assumptions understood, DO–178’s emphasis on software correctness is consistent with its stated purpose. Given that all the requirements necessary for ensuring adequate safety are eventually specified, then developing software that is correct with respect to those requirements is sufficient to ensure that the software does not negatively affect safety. Transforming safety into correctness is valid in this particular case.

As will be shown below, the e78-1.6 assurance case makes the transformation explicit. It also highlights the special role played by derived requirements.

3.1.2 Allowing life cycle flexibility

Another foundational concept of DO–178C 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 flexibility in choosing how to develop their software. Neither specific development methods nor life cycles are prescribed by the guidance. 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.’ (RTCA 2011b, p. 126)

The guidance does include detailed descriptions of specific activities that may be performed in order to satisfy particular objectives. References to the text of these activities are even included in the Annex A tables in 178C. 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. (RTCA 2011a, p. 3).

To emphasize the flexibility allowed by the guidance, the e78-1.6 assurance case does not explicitly include accomplishing any activities as goals that must be satisfied. Activities are only referenced within contextual items in the case.

3.1.3 Using confidence arguments

Researchers from the University of York and the University of Virginia (Hawkins et al. 2011) introduced the idea of a confidence argument to accompany a primary safety argument. The primary safety argument documents the arguments related to direct claims of safety; the confidence argument documents the arguments related to the sufficiency of confidence in the primary argument.

This separation into two different argument structures differs from the prevailing practice of intermixing concerns of safety and confidence in a single unified argument, and offers the potential promise of eliminating or mitigating some of the difficulties recognized in the prevailing approach (Haddon-Cave 2009). Although the original research concentrated on safety arguments, the general concept applies equally to any property of interest.

Even a cursory reading of DO–178C reveals that the guidance contains a mixture of objectives about the desired properties of the final software product, objectives related to intermediate products, and objectives concerning the processes used to develop the product. A more careful reading, keeping the notion of separating primary and confidence arguments in mind, suggests that some of these objectives naturally fit well into a primary argument about properties of the final software, and some naturally fit well into a confidence argument that affects the degree of belief in the sufficiency of the primary argument. Only a comparatively few objectives are difficult to classify.

These observations make using confidence arguments a foundational concept for the explicit assurance case. Reviewers of previous versions of the case commented favorably on this approach.

3.1.4 Explicating before evaluating

The fourth foundational concept is that accurately articulating the implicit case contained in DO–178C must precede trying to evaluate the sufficiency of the case. Evaluation is an important eventual goal of the research, but unless agreement can be reached about what the guidance really says, reaching agreement on whether it says the right thing is impossible.

The e78-1.6 assurance case is intended to properly capture what 178C says. Great effort was made to represent accurately the implicit arguments in the guidance, without trying to correct any perceived deficiencies. The coherence and cogency of this explicit case should be neither greater nor lesser than that of the guidance itself.

3.2 Characteristics of the case

The e78-1.6 assurance case expression in GSN consists of a primary argument module and a confidence argument module for each software level (A, B, C, D), generic primary and confidence argument modules, and a simple primary argument for software level E. Additional modules support the Level A-D primary and confidence arguments as follows:

- Level D
  - Primary argument: five supporting modules
  - Confidence argument: five support modules
- Level C:
  - Primary argument: two unique and two directly referenced level D supporting modules
  - Confidence argument: eight unique and five directly referenced level D supporting modules
- Level B:
  - Primary argument: one unique supporting module and a direct reference to the level C primary argument
  - Confidence argument: three unique, four directly referenced level C, and one directly referenced level D supporting modules
- Level A:
  - Primary argument: no unique supporting modules and a direct reference to the level B primary argument
  - Confidence argument: two unique, two directly referenced level B, two directly referenced level C, and one directly referenced level D supporting modules

Overall the 34 GSN modules for Levels A-D comprise 131 goals, 42 strategies, 176 context items, 34 justifications and assumptions, and 161 references to evidence. In some instances, the style of the GSN representation used in the project may rightly displease purists. Strict adherence to standard practices has been sacrificed in places under the belief that the sacrifice better achieves visual simplicity and enhances readability for the primary intended audience of the work, few of whom are experts in the notation.

Also, for the benefit of the intended audience, textual representations have been manually created for 15 of the GSN modules, with more in the works. For two of the five examples presented in the next section, a textual representation accompanies the GSN structure.

3.3 Excerpts from the case

Obviously the full case is too large to reproduce in this paper. Five representative excerpts are presented in this section: a simple version of the general primary argument, and one example each from the four main software levels.

3.3.1 Simple generic primary argument

Figure 1 shows a GSN representation of a very simple generic primary argument that captures the essence of the safety to correctness transformation, which, as noted above, constitutes the heart of the DO–178C implicit assurance case. It intentionally omits context, justifications, and assumptions for the sake of initial simplicity. These missing items do appear in the instantiation of the Level D primary argument shown in the next section.

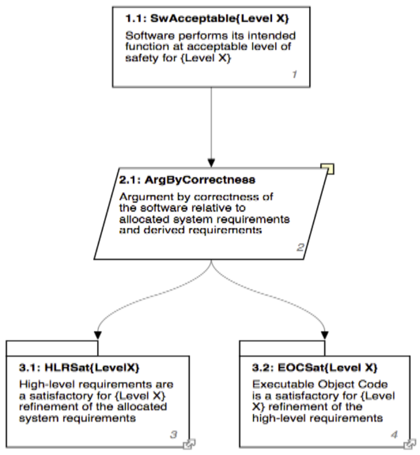
1.1: SwAcceptable(Level X)
Software performs its intended function at acceptable level of safety for {Level X}
1
2.1: ArgByCorrectness
Argument by correctness of the software relative to allocated system requirements and derived requirements
2
3.1: HLRSat(LevelX)
High-level requirements are a satisfactory for {Level X} refinement of the allocated system requirements
3
3.2: EOCSat(Level X)
Executable Object Code is a satisfactory for {Level X} refinement of the high-level requirements
4

Fig. 1. Simplified generic primary argument in GSN

Two aspects of the figure may be unclear to anyone unfamiliar with the particular tool set used in the project[3]. The number in the lower right hand corner of each element is a tool-generated unique identifier. It permits easy reference to a particular GSN element across an entire collection of arguments. The small appendage on the upper right corner of the ArgByCorrectness strategy element indicates a link to an associated confidence argument, which is contained in a separate GSN module.

A top-level primary argument for each software level D, C, B, and A could be expressed using an instantiation of this generic argument. In the e78-1.6 assurance case, the primary arguments for levels D (shown below) and C (not shown) are expressed in this way. The primary arguments for levels B and A are not, because using a different structure that highlights the specific ways these levels differ from the lower levels seemed more enlightening.

Using the structured textual format developed for the project, the simple generic argument may also be expressed as shown in Figure 2. Note that the text contained in item C within the '**if**' clause corresponds to the top-level goal of the associated confidence argument, which is not shown here.

The conclusion
    Software performs its intended function at acceptable level of safety for {level X}
is justified by an argument
    by correctness of the software relative to allocated system requirements and
    derived requirements
if
- A. High-level requirements are a satisfactory for {level X} refinement of the allocated system requirements; and
- B. Executable Object Code is a satisfactory for {level X} refinement of the high-level requirements; and
- C. The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for {Level X}

Fig. 2. Simplified generic primary argument in structured text

3.3.2 Level D primary argument

A GSN expression of the primary argument for software Level D is shown in Figure 3. Figure 4 presents an equivalent structured text representation of the same argument.

[3] The GSN structures were produced using tools created by Dependable Computing, Inc. Use of these tools does not imply an endorsement of them by the U.S. Government.

Text contained within double quotation marks is quoted directly from either DO–178C if no document is specified, or from the specified document otherwise. The location of the quotation is given in parentheses. For example, the text in MeaningAnomBeh comes from page 109 in the Glossary of DO–178C, and the text in HLRDev comes from Annex A table 2 row 1 of 178C. The text in 3.1 References comes from bullet 6 in section 5.4 of DO–248C. To keep the size of some elements reasonably small, quotations are not always given, but instead references to document locations are listed.

The Level D primary argument follows the structure illustrated in the previous section, but with appropriate context and assumptions added. Salient points about the argument include the following:

- The five context elements attached to the top-level claim in the GSN representation emphasize that the meaning of the claim can only be understood within an environment containing a description of the intended function of the software and definitions for acceptable level of safety, Level D, and anomalous behavior. Also, the top-level claim is relevant only for software that has been assigned to level D. In the textual representation, these pieces of information are identified as 'givens' when considering whether the desired conclusion holds.

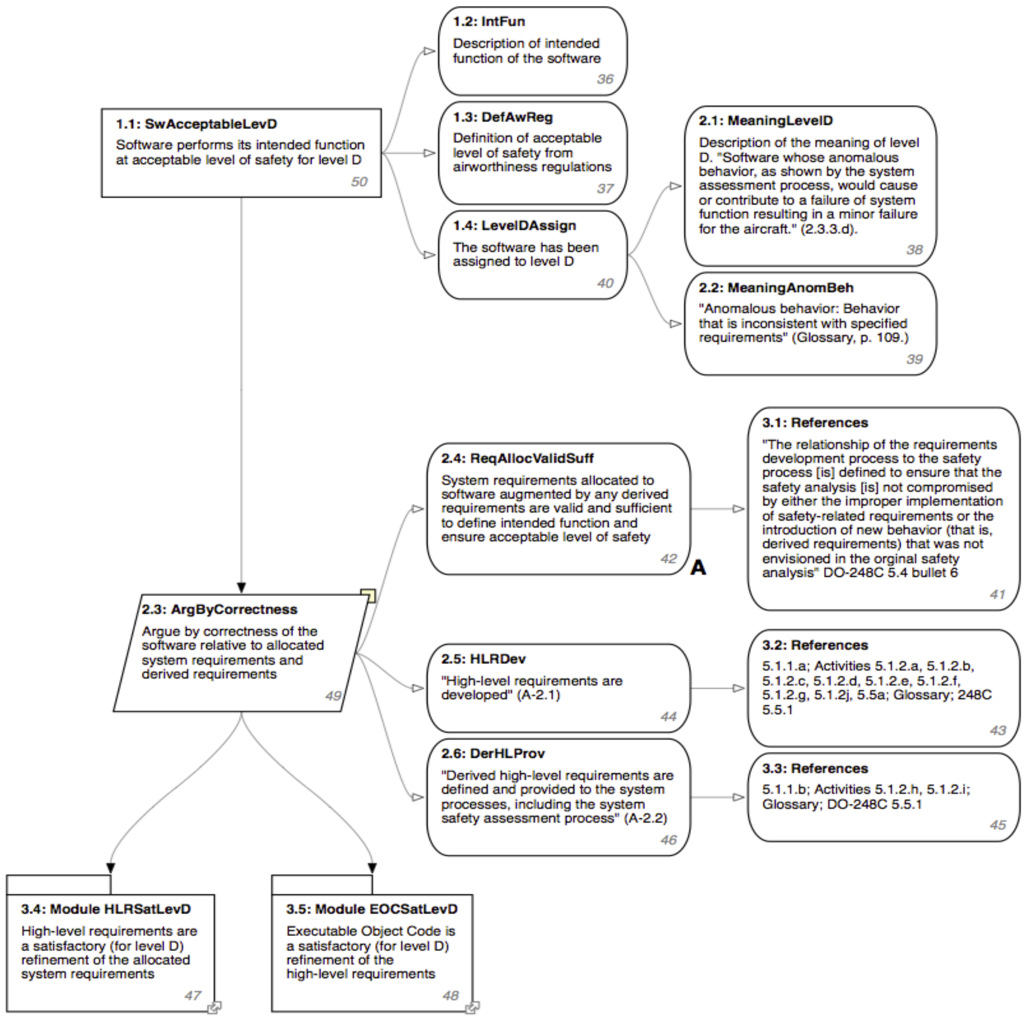
1.1: SwAcceptableLevD
Software performs its intended function at acceptable level of safety for level D
50
1.2: IntFun
Description of intended function of the software
36
1.3: DefAwReg
Definition of acceptable level of safety from airworthiness regulations
37
1.4: LevelDAssign
The software has been assigned to level D
40
2.1: MeaningLevelD
Description of the meaning of level D. "Software whose anomalous behavior, as shown by the system assessment process, would cause or contribute to a failure of system function resulting in a minor failure for the aircraft." (2.3.3.d).
38
2.2: MeaningAnomBeh
"Anomalous behavior: Behavior that is inconsistent with specified requirements" (Glossary, p. 109.)
39
2.4: ReqAllocValidSuff
System requirements allocated to software augmented by any derived requirements are valid and sufficient to define intended function and ensure acceptable level of safety
42
A
3.1: References
"The relationship of the requirements development process to the safety process [is] defined to ensure that the safety analysis [is] not compromised by either the improper implementation of safety-related requirements or the introduction of new behavior (that is, derived requirements) that was not envisioned in the orginal safety analysis" DO-248C 5.4 bullet 6
41
2.3: ArgByCorrectness
Argue by correctness of the software relative to allocated system requirements and derived requirements
49
2.5: HLRDev
"High-level requirements are developed" (A-2.1)
44
3.2: References
5.1.1.a; Activities 5.1.2.a, 5.1.2.b, 5.1.2.c, 5.1.2.d, 5.1.2.e, 5.1.2.f, 5.1.2.g, 5.1.2j, 5.5a; Glossary; 248C 5.5.1
43
2.6: DerHLProv
"Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process" (A-2.2)
46
3.3: References
5.1.1.b; Activities 5.1.2.h, 5.1.2.i; Glossary; DO-248C 5.5.1
45
3.4: Module HLRSatLevD
High-level requirements are a satisfactory (for level D) refinement of the allocated system requirements
47
3.5: Module EOCSatLevD
Executable Object Code is a satisfactory (for level D) refinement of the high-level requirements
48

Fig. 3. Level D primary argument in GSN

- The assumption ReAllocValidSuff explicitly identifies an essential part of the implicit assurance case within 178C. As discussed in section 3.1.1, the guidance is grounded in the belief that the requirements to which the software is built are sufficient to both fully define the intended function of the software and ensure achievement of an acceptable level of safety. The guidance itself does not directly justify this belief, but it does include objectives intended to ensure that the safety analysis processes are provided with adequate information to conduct a proper assessment.
- HLRDev and DerHLProv both refer to specific objectives in 178C. From the vantage point of the assurance case, these objectives seem more properly to establish the context in which the implicit correctness argument makes sense and satisfies the ReAllocValidSuff assumption than to identify propositions that must be shown to be true as part of the argument.

- HLRSatLevD and EOCSatLevD are the two prongs of the correctness argument. If the high-level requirements are a satisfactory refinement of the system requirements, and the executable object code is in turn a satisfactory refinement of these high-level requirements then the software can be said to be correct with respect to the allocated system requirements. By the safety to correctness transformation previously discussed, the software can therefore be said to perform its intended function at an acceptable level of safety for Level D.
- The associated confidence argument is not shown here, but its goal is identified in the textual representation as The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for level D.

The conclusion
  Software performs its intended function at acceptable level of safety for Level D
given
  A. Description of intended function of the software
  B. Definition of acceptable level of safety from airworthiness regulations
  C. The software has been assigned to Level D
  D. Description of the meaning of Level D: "Software whose anomalous behavior, as shown by the system assessment process, would cause or contribute to a failure of system function resulting in a minor failure condition for the aircraft for the aircraft." (2.3.3.d)
  E. "Anomalous behavior: behavior that is inconsistent with specified requirements" (Glossary, p. 109.)
is justified by an argument
  by correctness of the software relative to allocated system requirements and derived requirements
if
  A. High-level requirements are a satisfactory for Level D refinement of the allocated system requirements; and
  B. Executable Object Code is a satisfactory for Level D refinement of the high-level requirements; and
  C. The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for Level D
The argument assumes
  A. System requirements allocated to software augmented by any derived requirements are valid and sufficient to define intended function and ensure acceptable level of safety [see DO–248C 5.4 bullet 6]
  B. "High-level requirements are developed" (A-2.1) [see 5.1.1.a; Activities 5.1.2.a, 5.1.2.b, 5.1.2.c, 5.1.2.d, 5.1.2.e, 5.1.2.f, 5.1.2.g, 5.1.2j, 5.5a; Glossary; 248C 5.5.1]
  C. "Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process" (A-2.2) [see 5.1.1.b; Activities 5.1.2.h, 5.1.2.i; Glossary; DO–248C 5.5.1]

Fig. 4. Level D primary argument in structured text

3.3.3 Level C confidence argument

Thus far, confidence arguments have been mentioned several times, but none have been shown. Figure 5 remedies the situation by showing a GSN expression of the confidence argument for Level C software, slightly simplified to allow legible display on paper.

The goal of the confidence argument is to establish that the evidence used in the primary argument is adequate to justify believing that software correctness has been established. DO–178C's guidance related to showing the adequacy of processes for planning, configuration management, software quality assurance, verification of verification, and certification liaison all support gaining sufficient confidence. To enable Figure 5 to fit on the page, all of these are summarized in 3.1 5 Modules. In the full assurance case, separate modules exist related to each of the five processes.

To enhance confidence in the sufficiency of the two-level refinement process (system requirements to high-level requirements to executable object code), for Level C software, DO–178C introduces additional refinement steps, of which the guidance requires at least one (high-level to low-level), but allows for multiple in which 'the successive levels of requirements are developed such that each successively lower level satisfies its higher level requirements' (6.1.b). The possibility of multiple iterations of low-level requirements is denoted in the figure by the black circle on the arc from ArgEachRefinement to Module LLRSatLevC. The full assurance case includes details for each of the indicated modules.

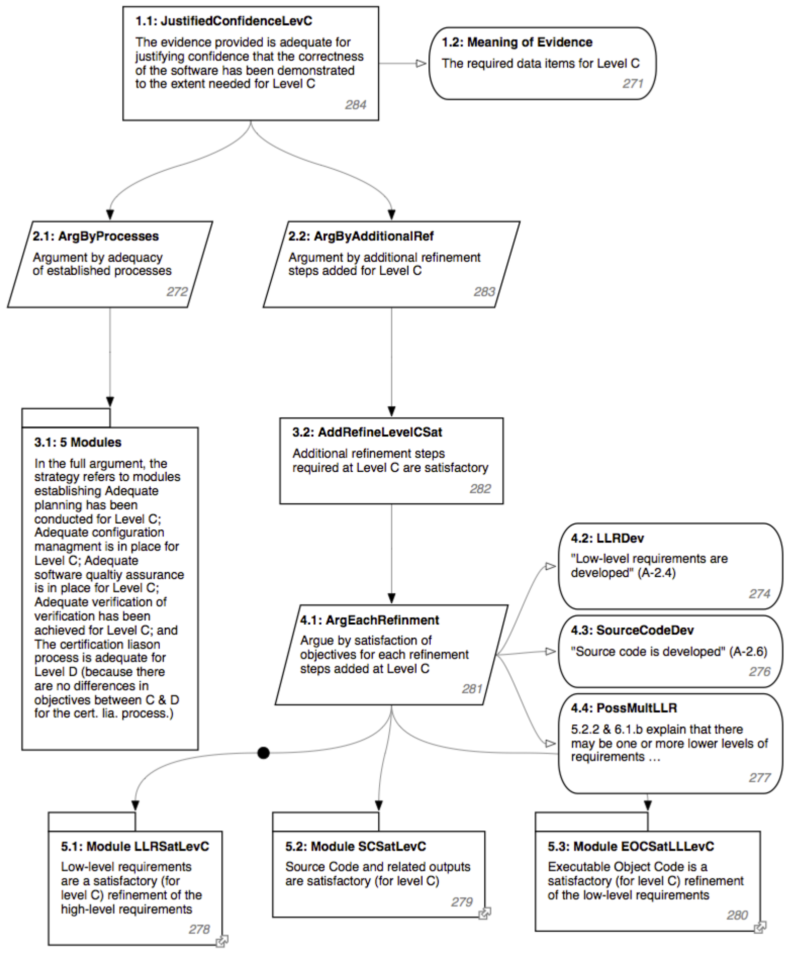
1.1: JustifiedConfidenceLevC
The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for Level C
284
1.2: Meaning of Evidence
The required data items for Level C
271
2.1: ArgByProcesses
Argument by adequacy of established processes
272
2.2: ArgByAdditionalRef
Argument by additional refinement steps added for Level C
283
3.1: 5 Modules
In the full argument, the strategy refers to modules establishing Adequate planning has been conducted for Level C; Adequate configuration managment is in place for Level C; Adequate software qualtiy assurance is in place for Level C; Adequate verification of verification has been achieved for Level C; and The certification liason process is adequate for Level D (because there are no differences in objectives between C & D for the cert. lia. process.)
3.2: AddRefineLevelCSat
Additional refinement steps required at Level C are satisfactory
282
4.2: LLRDev
"Low-level requirements are developed" (A-2.4)
274
4.1: ArgEachRefinment
Argue by satisfaction of objectives for each refinement steps added at Level C
281
4.3: SourceCodeDev
"Source code is developed" (A-2.6)
276
4.4: PossMultLLR
5.2.2 & 6.1.b explain that there may be one or more lower levels of requirements …
277
5.1: Module LLRSatLevC
Low-level requirements are a satisfactory (for level C) refinement of the high-level requirements
278
5.2: Module SCSatLevC
Source Code and related outputs are satisfactory (for level C)
279
5.3: Module EOCSatLLLevC
Executable Object Code is a satisfactory (for level C) refinement of the low-level requirements
280

Fig. 5. Level C confidence argument in GSN

3.3.4 Level B adequate planning argument

As an example from the Level B part of the e78-1.6 assurance case, Figure 6 shows the adequate planning component of the confidence argument.

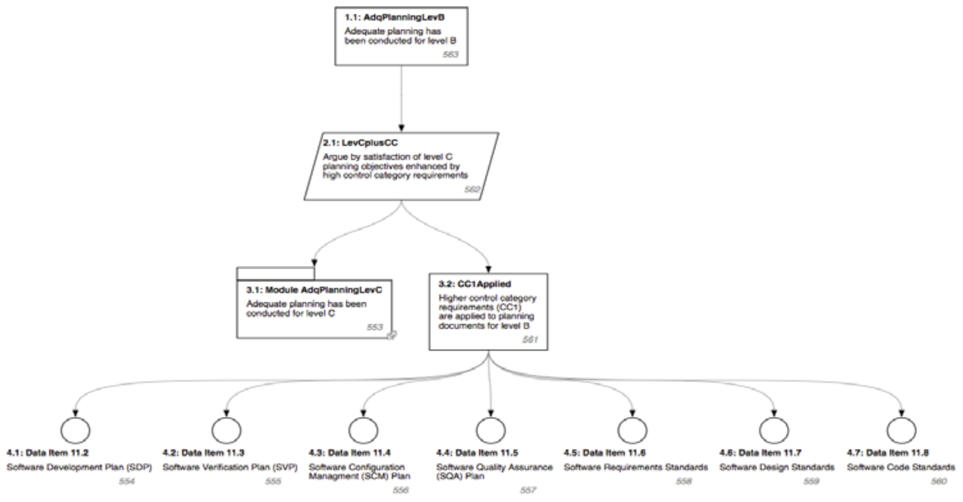
1.1: AdqPlanningLevB
Adequate planning has been conducted for level B
563
2.1: LevCplusCC
Argue by satisfaction of level C planning objectives enhanced by high control category requirements
562
3.1: Module AdqPlanningLevC
Adequate planning has been conducted for level C
553
3.2: CC1Applied
Higher control category requirements (CC1) are applied to planning documents for level B
561
4.1: Data Item 11.2
Software Development Plan (SDP)
554
4.2: Data Item 11.3
Software Verification Plan (SVP)
555
4.3: Data Item 11.4
Software Configuration Managment (SCM) Plan
556
4.4: Data Item 11.5
Software Quality Assurance (SQA) Plan
557
4.5: Data Item 11.6
Software Requirements Standards
558
4.6: Data Item 11.7
Software Design Standards
559
4.7: Data Item 11.8
Software Code Standards
560

Fig. 6. Level B adequate planning argument in GSN

The objectives for planning at Level B are the same as the objectives for Level C. The only difference lies in the raising of the control category that applies to the seven planning-related data items, which are shown here as evidence items.

3.3.5 Level A verification of verification process results argument

A final excerpt from the e78-1.6 assurance case is given in Figure 7. This structure constitutes the verification of verification process results module of the confidence argument for Level A.

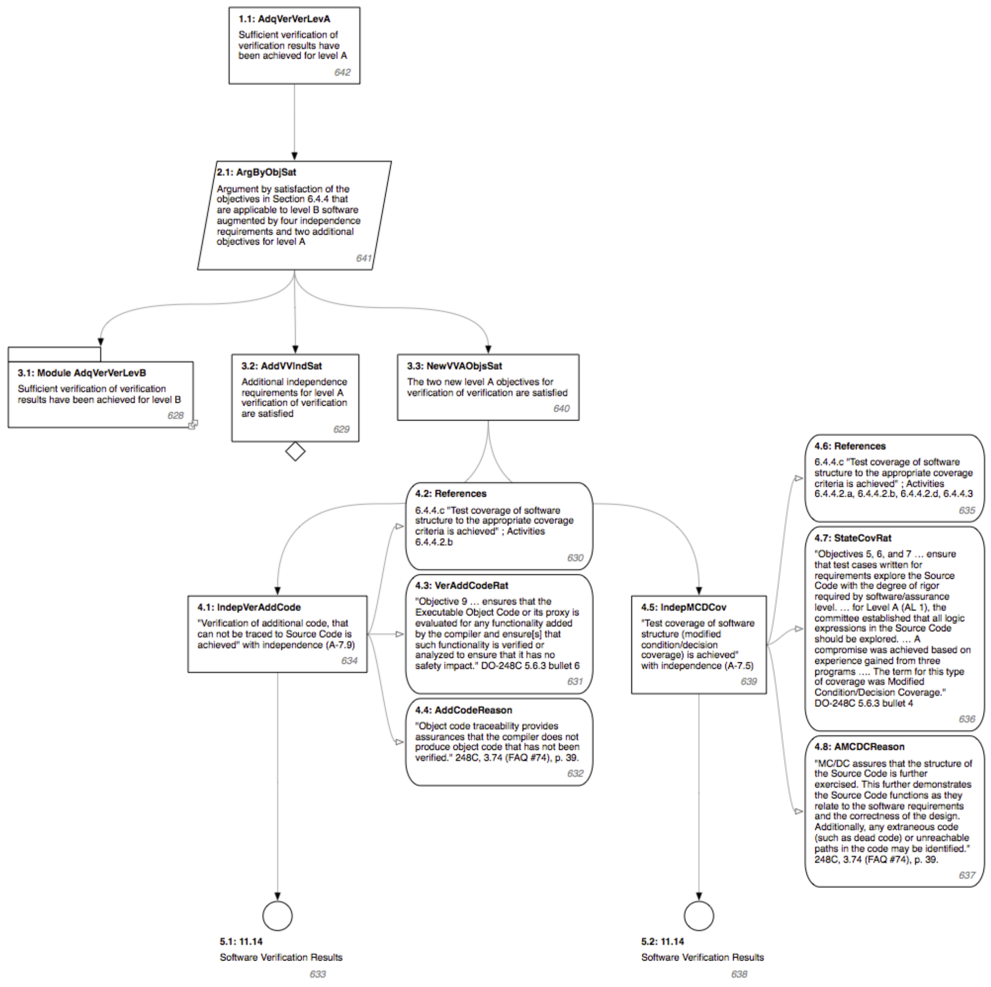
1.1: AdqVerVerLevA
Sufficient verification of verification results have been achieved for level A
642
2.1: ArgByObjSat
Argument by satisfaction of the objectives in Section 6.4.4 that are applicable to level B software augmented by four independence requirements and two additional objectives for level A
641
3.1: Module AdqVerVerLevB
Sufficient verification of verification results have been achieved for level B
628
3.2: AddVVIndSat
Additional independence requirements for level A verification of verification are satisfied
629
3.3: NewVVAObjsSat
The two new level A objectives for verification of verification are satisfied
640
4.2: References
6.4.4.c "Test coverage of software structure to the appropriate coverage criteria is achieved" ; Activities 6.4.4.2.b
630
4.1: IndepVerAddCode
"Verification of additional code, that can not be traced to Source Code is achieved" with independence (A-7.9)
634
4.3: VerAddCodeRat
"Objective 9 … ensures that the Executable Object Code or its proxy is evaluated for any functionality added by the compiler and ensure[s] that such functionality is verified or analyzed to ensure that it has no safety impact." DO-248C 5.6.3 bullet 6
631
4.4: AddCodeReason
"Object code traceability provides assurances that the compiler does not produce object code that has not been verified." 248C, 3.74 (FAQ #74), p. 39.
632
4.5: IndepMCDCov
"Test coverage of software structure (modified condition/decision coverage) is achieved" with independence (A-7.5)
639
4.6: References
6.4.4.c "Test coverage of software structure to the appropriate coverage criteria is achieved" ; Activities 6.4.4.2.a, 6.4.4.2.b, 6.4.4.2.d, 6.4.4.3
635
4.7: StateCovRat
"Objectives 5, 6, and 7 … ensure that test cases written for requirements explore the Source Code with the degree of rigor required by software/assurance level. … for Level A (AL 1), the committee established that all logic expressions in the Source Code should be explored. … A compromise was achieved based on experience gained from three programs …. The term for this type of coverage was Modified Condition/Decision Coverage." DO-248C 5.6.3 bullet 4
636
4.8: AMCDCReason
"MC/DC assures that the structure of the Source Code is further exercised. This further demonstrates the Source Code functions as they relate to the software requirements and the correctness of the design. Additionally, any extraneous code (such as dead code) or unreachable paths in the code may be identified." 248C, 3.74 (FAQ #74), p. 39.
637
5.1: 11.14
Software Verification Results
633
5.2: 11.14
Software Verification Results
638

Fig. 7. Level A verification of verification process results argument in GSN

Verification of verification process results for Level A differs from Level B only in having additional requirements for independence (which are not elaborated in the figure, but are in the full assurance case), and two new objectives: verifying untraceable code (IndepVerAddCode) and achieving modified condition / decision coverage (IndepMCDCov), which must be done with independence.

4 Next steps and concluding remarks

The e78-1.6 assurance case discussed in this paper is not the final product of the Explicate '78 research. The case needs to be subjected to careful scrutiny by aviation industry and regulator experts, as well as assurance case and GSN experts. For the former, the existing textual representations most likely will need to be expanded to include the entire case. For the latter, the somewhat loose use of GSN elements that characterize the current case will likely need to be tightened.

The current case, however, seems to be sufficiently stable and complete to permit two concurrent activities to be undertaken during the heightened scrutiny period:

1. Beginning to evaluate the sufficiency of the case, not just as an accurate reflection of what DO–178C requires, but also as to whether what it requires is strong enough at each software level to provide justified assurance that software that complies with the document will perform 'its intended function with a level of confidence in safety that complies with airworthiness requirements.

2. Extending the existing case to include the guidance from one or more of the supplement documents.

If all goes well, good progress on all of these activities will be made before this paper is published. The goal is to complete the research before the end of 2015.

At least four benefits may arise from successful completion of this research, two of which are specific to DO–178C, and two of which are more general. First, the existence of an explicit assurance case for the DO–178C guidance should facilitate intelligent conversations about the relative efficacy of DO–178C and proposed alternative approaches for demonstrating compliance with airworthiness regulations. The likelihood of this benefit truly happening increases with the number of people within industry and the regulatory authorities who accept the Explicate '78 assurance case as an accurate reflection of the guidance.

Second, effectively analysing the adequacy of the assurance case should provide a solid foundation for future modifications to the guidance. When the time comes to create DO–178D, perhaps the Explicate '78 results will help provide the committee with a more structured and systematic basis for making changes than an unordered list of issues.

Third, more generally the existence of an assurance case representation for one guidance document may motivate the creation of such representations for other guidance documents. This, in turn, may result in clearer understanding of and more systematic updates to such documents.

Fourth, and most generally of all, perhaps the Explicate '78 work may help serve as a catalyst for prompting improved cooperation and mutual understanding between supporters of prescriptive standards and supporters of goal-based standards. One might even go so far as to hope for a lasting peace.

Acknowledgments This work is partially funded by the Reimbursable Interagency Agreement DTFACT-10-X-00008, Modification 0004, Space Act IAI-1073, between the Federal Aviation Administration and the National Aeronautics and Space Administration, Langley Research Center, for Design, Verification, and Validation of Advanced Digital Airborne Systems Technology: Annex 2, Assurance Case Applicability to Digital Systems. This paper, however, did not undergo any official review by FAA personnel.

References

Ankrum T, Kromholz A (2005) Structured Assurance Cases: Three Common Standards. Proceedings of the Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05). Heidelberg, Germany

Ayoub, A, Chang, J, Sokolsky, O, & Lee, I (2013) Assessing the Overall Sufficiency of Safety Arguments. Assuring the Safety of Systems: Proceedings of the Twenty-first Safety Critical Systems Symposium. C. Dale & T. Anderson (Eds.). February 5–7. Bristol, UK. Springer

Bloomfield R, Bishop P (2010) Safety and Assurance Cases: Past, Present and Possible Future. Making Systems Safer. C. Dale and T. Anderson (eds). Springer-Verlag

Denney, E, Pai, G, Habli, I, Kelly, T, & Knight, J (2013). 1st International Workshop on Assurance Cases for Software-intensive Systems (ASSURE 2013). Proceedings of the 2013 International Conference on Software Engineering. May 18–26. San Francisco, California.

European Aviation Safety Agency (2013) AMC 20-115C Software Considerations for Certification of Airborne Systems and Equipment. ED Decision 2013/026/R http://easa.europa. eu/system/files/dfu/Annex%20II%20-%20AMC%2020-115C.pdf (last accessed December 2, 2014)

Federal Aviation Administration (2013a) Standard Airworthiness Certification: Regulations – Title 14 Code of Federal Regulations. http://www.faa.gov/aircraft/air_cert/airworthiness_ certification/std_awcert/std_awcert_regs/regs/ (last accessed December 5, 2014)

Federal Aviation Administration (2013b) Advisory Circular 20-115C Airborne Software Assurance. http://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_20-115C.pdf (last accessed December 2, 2014)

Galloway A, Paige R, Tudor, N, Weaver R, McDermid, J. (2005) Proof vs. Testing in the Context of Safety Standards. The 24th Digital Avionics Systems Conference (DASC), Washington D.C.

Goodenough J, Weinstock C, Klein A (2012) Toward a Theory of Assurance Case Confidence. CMU-SEI-2002-TR-002, September

Graydon, P (2014) Towards a Clearer Understanding of Context and Its Role in Assurance Argument Confidence. Computer Safety, Reliability, and Security, 139-154.

GSN Committee (2011) Draft GSN Standard Version 1.0. http://www.goalstructuringnotation. info/ (last accessed December 2, 2014)

Haddon-Cave C (2009) The Nimrod Review. London: The Stationary Office http://www.official-documents.gov.uk/document/hc0809/hc10/1025/1025.pdf (last accessed December 1, 2014).

Hawkins R, Habli I, Kelly T, McDermid J (2013) Assurance cases and prescriptive software safety certification: A comparative study. Safety Science. Vol 59

Hawkins R, Kelly T (2009) A Systematic Approach for Developing Software Safety Arguments. Proceedings of the 27th International System Safety Conference. Huntsville, Alabama

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

Holloway CM (2013) Making the Implicit Explicit: Towards an Assurance Case for DO–178C. Proceedings of the 31st International System Safety Conference. August 12-16. Boston, Massachusetts (ref. z)

Holloway CM (2012) Towards Understanding the DO–178C / ED–12C Assurance Case. 7th IET International Conference on System Safety, Incorporating the Cyber Security Conference. Edinburgh

Holloway CM (2008) Safety Case Notations: Alternatives for the Non-Graphically Inclined? Proceedings of the 3rd IET International System Safety Conference. Birmingham, UK

Knight J (2012) Fundamentals of Dependable Computing for Software Engineers. Boca Raton, Florida: CRC Press

Matsuno, Y (2014) A Design and Implementation of an Assurance Case Language. Dependable Systems and Networks (DSN). Atlanta, Georgia

RTCA (1992) Software Considerations in Airborne Systems and Equipment Certification. DO–178B.

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

RTCA (2011b) Supporting Information for DO–178C and DO–278A. DO–248C

RTCA (2011c) Software Integrity Assurance Considerations for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems. DO–278A

RTCA (2011d) Software Tool Qualification Considerations. DO–330

RTCA (2011e) Model-Based Development and Verification Supplement to DO–178C and DO–278A. DO–331

RTCA (2011f) Object-Oriented Technology and Related Techniques Supplement to DO–178C and DO–278A. DO–332

RTCA (2011g) Formal Methods Supplement to DO–178C and DO–278A. DO–333

Rushby, J (2013) Logic and epistemology in safety cases. Computer Safety, Reliability, and Security, 32nd SAFECOMP. Toulouse, France

Rushby J (2011) New Challenges in Certification of Aircraft Software. Proceedings of the 11th International Conference on Embedded Software (EMSOFT). Taipei, Taiwan

SAE International (1996) Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment. SAE ARP 4761

SAE International (2010) Guidelines for Development of Civil Aircraft and Systems. SAE ARP 4754a

Toulmin S (2003) The Uses of Argument, Updated Edition. Cambridge University Press

UK Ministry of Defence (2007) Defence Standard 00-56 Issue 4: Safety Management Requirements for Defence Systems

Yuan T, Kelly T (2011) Argument Schemes in Computer System Safety Engineering. Informal Logic 31 (2)