**DOT/FAA/TC-17/67**

Federal Aviation Administration
William J. Hughes Technical Center
Aviation Research Division
Atlantic City International Airport
New Jersey 08405

# Explicate ’78: Assurance Case Applicability to Digital Systems

January 2018

Final Report

U.S. Department of Transportation
**Federal Aviation Administration**

# NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof. The U.S. Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report. The findings and conclusions in this report are those of the author(s) and do not necessarily represent the views of the funding agency. This document does not constitute FAA policy. Consult the FAA sponsoring organization listed on the Technical Documentation page as to its use.

This report is available at the Federal Aviation Administration William J. Hughes Technical Center's Full-Text Technical Reports page: actlibrary.tc.faa.gov in Adobe Acrobat portable document format (PDF).

## Technical Report Documentation Page

| | | |
|---|---|---|
| 1. Report No.<br>DOT/FAA/TC-17/67 | 2. Government Accession No. | 3. Recipient's Catalog No. |

| | |
|---|---|
| 4. Title and Subtitle<br>EXPLICATE '78: ASSURANCE CASE APPLICABILITY TO DIGITAL SYSTEMS | 5. Report Date<br>January 2018 |
| | 6. Performing Organization Code |
| 7. Author(s)<br>C. Michael Holloway & Patrick J. Graydon | 8. Performing Organization Report No. |
| 9. Performing Organization Name and Address<br>NASA Langley Research Center, 100 NASA Road, Hampton VA 23681 | 10. Work Unit No. (TRAIS) |
| | 11. Contract or Grant No.<br>IAI-1073 |
| 12. Sponsoring Agency Name and Address<br>Federal Aviation Administration<br>William J. Hughes Technical Center<br>Aviation Research Division<br>Atlantic City International Airport, NJ 08405 | 13. Type of Report and Period Covered<br>Final Report |
| | 14. Sponsoring Agency Code<br>Barbara Lingberg, AIR-6B4 |

15. Supplementary Notes

The FAA William J. Hughes Technical Center Aviation Research Division Technical Monitors were Charles Kilgore and Srini Mandalapu.

16. Abstract

This report documents the results of the Explicate '78 project. The project was conducted by NASA Langley Research Center in support of an annex (Assurance Case Applicability to Digital Systems) to the Reimbursable Interagency Agreement IA1-1073 (Design, Verification, and Validation of Advanced Digital Airborne Systems Technology). In particular, the report describes an assurance case developed to express the arguments contained in, or implied by, DO-178C (Software Considerations in Airborne Systems and Equipment Certification), which implicitly justifies the assumption that the document meets its stated purpose of providing "guidelines 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." An appendix to the report provides an assurance case for DO-330 (Software Tool Qualification Considerations).

| | |
|---|---|
| 17. Key Words<br>DO-178C, Assurance case, Argument, Software, Correctness, Safety, DO-330 | 18. Distribution Statement<br>This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161. This document is also available from the FAA William J. Hughes Technical Center at actlibrary.tc.faa.gov. |

| | | | |
|---|---|---|---|
| 19. Security Classif. (of this report)<br>Unclassified | 20. Security Classif. (of this page)<br>Unclassified | 21. No. of Pages<br>191 | 22. Price |

# ACKNOWLEDGEMENTS

Without the initial encouragement of Mike DeWalt (the retired Chief Scientist and Technical Advisor for Software), the Explicate ’78 project would not have taken place. Without his careful review of the work at several stages, the results would not have continued. Barbara Lingberg also contributed important technical insights. Chuck Kilgore (now retired) and Srini Mandalapu provided helpful program management support, particularly in accommodating schedule changes necessitated by a variety of unexpected events that transpired during the project. To all of you, thanks!

## TABLE OF CONTENTS

APPENDICES

# LIST OF FIGURES

# LIST OF ACRONYMS

| | |
|---|---|
| CNS/ATM | Communication, Navigation, Surveillance and Air Traffic Management |
| EUROCAE | European Organisation for Civil Aviation Equipment |
| GSN | Goal Structuring Notation |
| TQL | Tool Qualification Level |

# EXECUTIVE SUMMARY

The Explicate '78 project was conducted by NASA Langley Research Center in support of an annex (Assurance Case Applicability to Digital Systems) to a Reimbursable Interagency Agreement IA1-1073 (Design, Verification, and Validation of Advanced Digital Airborne Systems Technology) between NASA Langley Research Center and the FAA.

This report documents two of the main achievements of the Explicate '78 research:

1. Expressing, as an assurance case, the arguments contained in, or implied by, DO-178C, which implicitly justifies the assumption that the document meets its stated purpose of providing "guidelines 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"
2. Expressing as an assurance case the arguments contained in, or implied by, DO-330, whose stated purpose "is to provide tool qualification guidance"

Substantial portions of the DO-178C assurance case are presented and explained in the body of the report, with the entire case presented in appendix A. Brief but substantive explanatory materials about DO-178C (and associated documents) and about assurance cases, evaluative observations and analysis, and representative arguments from the technology supplements are also presented in the body of the report. The complete DO-330 assurance case is presented and explained in appendix B. Previous papers written about the work are reprinted in appendix C.

## 1. INTRODUCTION

In September 2012 representatives from NASA Langley Research Center and the FAA signed an annex (Assurance Case Applicability to Digital Systems) to the Reimbursable Interagency Agreement IA1-1073 (Design, Verification, and Validation of Advanced Digital Airborne Systems Technology). The annex initiated research to create an assurance case framework for the guidance document DO-178C: Software Considerations in Airborne Systems and Equipment Certification [1], and to develop educational materials and argument evaluation criteria. The research collectively came to be called Explicate '78.

The specific activities agreed to be undertaken included the following:

- Expressing, as an assurance case, the arguments contained in, or implied by, DO-178C, which implicitly justifies the assumption that the document meets its stated purpose of providing "guidelines 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"
- Determining whether there is a need to conduct an analysis similar to the DO-178C analysis for any of the four supplementary documents associated with DO-178C, and, conducting any such analysis deemed worthwhile
- Providing educational materials about the basic principles, terminology, and existing uses of assurance/safety cases
- Developing argument evaluation criteria for determining whether an assurance case is sufficient for purpose

This report fully documents the first two of these activities, while making use of results from the third and fourth activities where appropriate. Full documentation of those two activities has been previously provided to the FAA in the form of a separate series of video presentations and transcripts [2–6], augmented by two NASA contractor reports [7, 8].

Deciding how to best organize the report was difficult. Specific difficulties included determining how much background information to provide and in what form to provide it; selecting the style and order of presentation for the individual elements of the DO-178C arguments; choosing the depth, breadth, and detail of evaluation comments to present; and deciding how to present the analysis of the chosen supplementary document. Many different potential solutions to these difficulties were tried and found wanting. The solutions adopted in this report are not perfect but should be adequate for the most likely readers of the report.

- Background information is provided at a fairly high level, with references given for the benefit of readers who may need more details.
- The DO-178C arguments are presented in a graphical style nearly identical to the popular Goal Structuring Notation (GSN) [9], with textual commentary added for further explanation.
- The order of explication of the DO-178C arguments mimics the order in which the arguments were originally created, namely beginning with the least critical software level to which the document applies (Level D) and proceeding level-by-level to Level A.

Only a few selected arguments are presented in the main body of the report. The full collection is contained in appendix A.

- Evaluation results are summarized and explained in a separate section.
- The analysis of the chosen supplementary document (DO-330: Software Tool Qualification Considerations [10]) is presented in appendix B. A brief explanation of why it was chosen for analysis, and why the other three supplements were not, is included in the main body of this report.
- The three previously published conference papers about Explicate '78 [11–13]) are reproduced in full in appendix C. Some material from these papers is used verbatim in this report. Some other material was made obsolete as the project progressed.

## 2. BACKGROUND

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

### 2.1 ABOUT DO-178C

For the benefit of the readers who are not familiar with DO-178C, a short discussion of the DO-178C's history is provided in this section. The information relies heavily 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 behavior 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 the European Organisation for Civil Aviation Equipment (EUROCAE)[1] 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 include an "objective-based approach for software assurance" and the "technology independent nature" of the objectives. The special committee/working group was also directed to maintain "backward compatibility with DO- 178B" except where doing so would fail to

[1] At one time, RTCA was an abbreviation for Radio Technical Commission for Aeronautics; since 1991 the four letters have been the freestanding name of the organization. EUROCAE uses a different document numbering scheme, but the content of the documents is otherwise identical. For example, DO-178C is called ED–12C. Only the DO numbering is used in this report.

"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, which are DO-278A: Software Integrity Assurance Considerations for Communication, Navigation, Surveillance and Air Traffic Management (CNS/ATM) Systems [14] and DO-248C: Supporting Information for DO-178C and DO-278A [15]. 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 a brief rationale.

Four new documents were also published to address specific issues and techniques: DO-330: Software Tool Qualification Considerations [10]; DO-331: Model-Based Development and Verification Supplement to DO-178C and DO-278A [16]; DO-332: Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A [17]; and DO-333: Formal Methods Supplement to DO-178C and DO-278A [18]. The general 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 rewrite 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 previously enumerated for formal methods, object-oriented technology, and model-based design and verification.

The relevant documents received official regulatory authority recognition in 2013 [19, 20].

## 2.2 ABOUT ASSURANCE CASE PRINCIPLES

The concept of an assurance case is a generalization of the safety case concept. A common definition[2] of a safety case is "a structured argument, supported by a body of evidence that provides a … case that a system is safe for a given application in a given operating environment" [21]. Conclusions are made concerning the achievement of an acceptable level of safety, and arguments are focused on providing justified confidence that those safety conclusions are satisfied. A more general *assurance case* concerns providing justified confidence about desired attributes in addition

[2] The elided part of the quoted definition includes the adjectives "compelling, comprehensible and valid." With those adjectives, the definition embeds notions of goodness, which is inappropriate in a definition for a phrase that may be applied to something that only purports to be a good safety case, but which may, after evaluation, be found to be neither compelling, comprehensible, nor valid. The main body of 00-56, issue 6, implicitly recognizes this distinction, but the definition has not been changed.

to safety, such as correctness, functionality, performance, or security. Hereafter in this report, this general term will be used.

Claims, arguments, evidence[3], context, and assumptions constitute five components of a well-structured assurance case [22]. A *claim*, within the assurance case literature, refers to a constative statement about some attribute or aspect related to the system being considered. *Goal* is another common name for the same concept within the assurance case community. Two terms commonly used for centuries within the philosophy and logic communities for the same concept are *proposition* and *conclusion* [23, 24]. The term *conclusion* is used in this report because it tends not to have any of the negative connotations of the other terms.

In a full assurance case, there will likely be many conclusions that must be shown to hold[4] at varying levels of generality. An example of a high-level conclusion is: **The software performs its intended function at an acceptable level of safety** (the sentences in **bold** throughout the report denote assurance case text). Examples of conclusions with increasing levels 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.**

In the assurance case literature, the term *argument* is overloaded. It is commonly used to refer to the overall case or to portions of the case that can be described separately. It is also commonly used in a more restrictive sense of that which explains how a conclusion is supported by, or justifiably inferred from, the evidence and associated lower-level conclusions. Perhaps to partially alleviate the overloading confusion, the GSN uses the term *strategy* for this latter situation. The most appropriate term, however, is *warrant* [25], which is used in this report.

*Evidence*, as commonly used in the assurance case community, refers to the available body of known facts relevant to the case being considered. *Data*, *fact*, and *solution* are synonymous terms. Examples of evidence include hazard logs, testing results, and mathematical theorems. When considered in light of traditional treatments of argumentation, evidence is nothing more than a special type of *premise*. In such treatments, a premise is a statement cited in support of a conclusion, which it is presumed the listener or reader will readily accept as true (either immediately, or as the result of another argument supporting its truth). Both *premise* and *evidence* are used in this report. The former is generally used for statements in which additional argument is provided to justify their truth; the latter is used for premises at the "bottom" of an argument structure.

[3] The claims, argument, and evidence distinction are established within the safety case literature. The terms and distinctions are briefly addressed in this section.

[4] The phrase "shown to be true" and variants, such as "shown to hold," are implicitly modified by "… to an appropriate degree of confidence for the case under consideration." Truth is of concern, in the practical, engineering sense, but not in an absolute philosophical or theological sense.

*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. For example, the context for a conclusion “**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 [26], 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 [27]. The looser, more common notion of context is used in this report.

An *assumption* is a statement on which the case relies but which is not elaborated or argued for in the assurance case. It is simply assumed to hold. As an example, an argument concerning safety that concludes that **all identified hazards have been eliminated may rely on the assumption that all credible hazards have been identified**.

These five concepts, no matter which terms are used to express them, 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. The use of assurance cases is not a new way of engineering, but rather a way of documenting in arguments what engineers already do. An active research community is exploring how to best create, express, analyze, improve, and maintain assurance cases. Readers interested in learning more about assurance case research are encouraged to explore the references cited in the papers reproduced in appendix C of this report, along with other references cited directly in the main body.

One additional assurance case concept—the confidence argument—plays an important role in the DO-178C case. The idea of separating primary and confidence arguments was first introduced in 2011 by researchers from the Universities of Virginia and York [28]. It involves distinguishing between arguments making direct conclusions about the attributes of interest (e.g., safety in a safety case) and arguments related to sufficiency of confidence instead of intermingling these different concerns into a single-argument structure. The Explicate ‘78 research did not apply the specific proposed mechanisms, but it did make extensive use of the general concept. This general concept is especially appropriate for DO-178C. Even a cursory reading of the guidance reveals that it contains a mixture of objectives about the desired properties of the final software product, intermediate products, and the processes used to develop the product. With only a few exceptions, this mixture separates rather cleanly into primary and confidence arguments.

## 2.3 ABOUT THE GSN

The GSN is a popular graphical notation for expressing assurance cases. For developing GSN diagrams for Explicate ‘78, a set of tools from Dependable Computing, Inc. (DCI) was used. Some of the primary symbols of the notation as rendered by the tools are shown in figure 1. The standard GSN names for the concepts are used in the figure. However, *conclusion* is used instead of *goal* and *warrant* instead of *strategy*. The standard GSN uses ellipses, not rounded rectangles, for assumptions and justifications, as is used by the DCI tools. Text within these symbols is used to provide content and a convenient means of referring to individual elements.

The concepts represented by most of these elements have already been described. The new concepts introduced in the figure are as follows. A *justification* gives the rationale for why a particular strategy (warrant) or goal (conclusion) is acceptable. A *module* provides a means for referring to a claim (conclusion) that is elaborated in a separate argument. To construct an argument, the elements of the GSN notation are linked together using the *in context of* or *supported by* directed lines. The *undeveloped entity* symbol is appended to the bottom of another element to indicate that the particular line of argument requires further development.

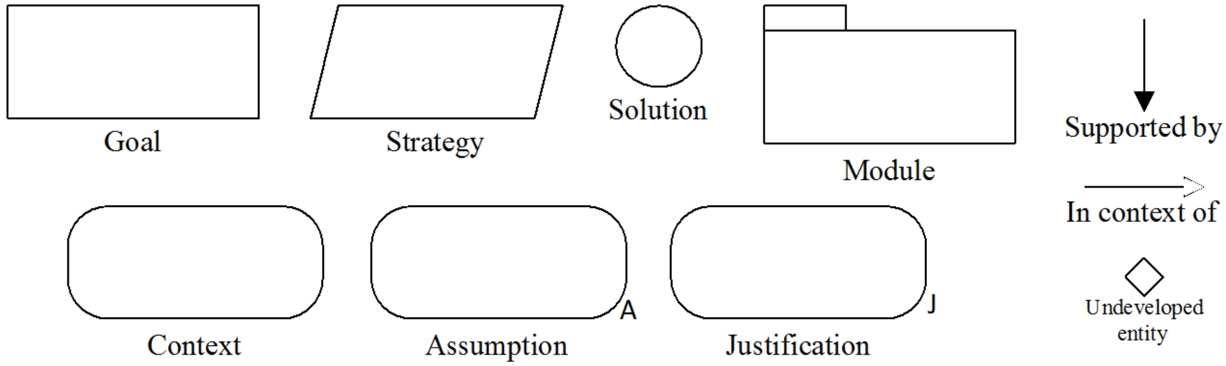
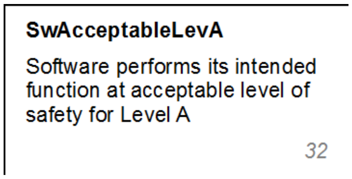


**Figure 1. Some Elements of GSN**

The meaning of two or more *supported by* arrows proceeding from a goal (conclusion) or strategy (warrant) is that all elements pointed to by the arrows are necessary to provide support. Also, all elements at a lower level in the structure inherit the context and assumptions attached to their ancestors in the structure.[5]

For the implementation of GSN used in Explicate '78, the text within each graphical element consists of three parts, one numerical and two alphabetical, as shown in figure 2.



**SwAcceptableLevA**

Software performs its intended function at acceptable level of safety for Level A

32

**Figure 2. Example of a GSN Element**

The number towards the lower right-hand side (32 in the example) is a unique identifier that is used in producing a comprehensive index. It is called the GSN ID. The GSN ID distinguishes among GSN elements across the entire collection of arguments. Whether GSN IDs are generated is controlled by the argument developer. These are present in all of the DO-178C arguments developed for Explicate '78.

[5] This statement is not true of modules in standard GSN. For reasons of simplicity and readability, standard GSN practices have not been strictly adhered to. The away goal and pattern constructs have not been used.

The two alphabetical parts are written when developing the argument. The bold text at the top (SwAcceptableLevA in the example) simply serves as an identifier; it is optional, but used throughout the Explicate ‘78 arguments. The normal text in the middle of the graphical element is the essential content: **Software performs its intended function at acceptable level of safety for Level A** in the example.

The DCI tools used in Explicate ‘78 also provide support for two forms of embedded links:

- Module link: denoted by a small square with an arrow on the bottom right-hand corner of an element. This link only appears in electronic versions of the arguments; it does not appear in printed versions.
- Confidence link: denoted by a small yellow square partially hidden behind the upper right hand corner of an element. This link appears in both electronic and printed versions.

Other aspects of the notation are explained below when they first appear.

## 2.4 ABOUT 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 [29]. 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 conclusion was that DO-178B Software Considerations are taken into account. Sub-conclusions were given for each of the DO-178B chapters (2–9); for example: 2.0 System Aspects were taken into account; the 5.0 Software Development Process was executed as planned; and the 9.0 Certification Liaison process was properly established and executed.

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 Explicate ‘78 research, which concentrated 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. 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.

## 2.5 SUMMARY OF TERMS

The following list enumerates and explains the specific terms that are used in the description of the DO-178C assurance case:

- **argument**: a structure consisting of a conclusion, one or more premises, and a warrant, along with possible additional information in the form of context and assumptions. The purpose of an argument is to convince the reader that its conclusion is true. Recall that "true" in the context of this report is assumed to always be modified by "to an appropriate degree of confidence for the case under consideration."
- **conclusion**: a statement whose truth is asserted as a consequence of the warrant and premises
- **warrant**: an explanation of the reason(s) the premises are sufficient to establish the truth of the conclusion
- **premise**: a statement that, if true, contributes positively to the truth of the conclusion. Some premises may themselves be conclusions of supporting arguments.
- **evidence**: a special form of premise that identifies known facts. No additional argument is needed for evidence.
- **context**: additional information needed to clarify or constrain the meaning of parts of an argument
- **assumption**: a statement whose truth is necessary for the conclusion to hold, but for which no additional argument or elaboration is provided
- **main argument**: an argument making direct conclusions about the attributes of interest in the case; may also be called a **top-level argument**
- **confidence argument**: an argument concerning whether confidence is justified in relevant aspects of the main argument

# 3. THE IMPLICIT ASSURANCE CASE IN DO-178C

With the preceding background information as a foundation, it is now easier to discuss the assurance case developed to describe the guidance in DO-178C. Throughout this discussion, the definite pronoun will be employed when referring to this assurance case. This usage is not intended to imply that the specific case developed in this research is the only, or even necessarily the best, case that can be developed. As indicated in previous work, several alternative approaches were explored, and others briefly considered. Pages C-24 through C-27 of appendix C present the early steps taken in discovering and developing this case.

## 3.1 GUIDING PRINCIPLES

Three principles guided the creation of the DO-178C assurance case: 1) faithfulness to the text; 2) minimum speculation; and 3) explication before evaluation. These principles were adopted to help guard against straying from the purpose of the research, which, as indicated previously, was to accurately represent the implied assurance case in DO-178C. The possibility was therefore reduced of inadvertently (or intentionally), creating a case that conformed more to a personal concept of an ideal case than to the guidance.

Maintaining faithfulness to the text dictated using actual words from the guidance whenever possible. For example, whenever an argument element represented a specific objective, the full text of the objective was reproduced in the argument along with citations to both the text section in which it appears and the associated Annex table entry. Faithfulness to the text also required consulting the explanatory material in DO-248C to clarify possible ambiguities.

In one area, following the faithfulness to the text principle produced some surprising results. Except for a handful of instances, nothing about activities appears in the arguments. DO-178C section 1.4.d explicitly states that an applicant "may plan and … adopt alternative activities to those described in this document."[6] Thus, the activities described in the guidance are only suggestions, not an integral part of the implied assurance case.

The primary application of the minimum speculation principle was for explicating the warrants associating premises and conclusions. In many instances, neither the DO-178C guidance nor the DO-248C additional information contained any explicit or strongly implied reasons explaining why certain premises should be considered to justify a particular conclusion. For such instances, the warrant included in the argument took a trivial compositional form.

The explication before evaluation principle is almost self-explanatory. The full set of arguments was created before any evaluation of their sufficiency was undertaken. However, throughout the creation of the arguments, periodic assessment was conducted regarding whether they accurately captured the reasoning contained in the guidance. The explication before evaluation principle was also followed in organizing the rest of the current section. A full representative sample of the arguments is presented before any evaluative comments are made about them.

## 3.2 LEVEL D ARGUMENTS

All the Level D arguments are shown in this section, with varying amounts of descriptive text, beginning with much detail and decreasing throughout the section in anticipation that the reader will grow accustomed to the content and style of the graphical arguments[7].

The top-level assurance argument (see figure 3) establishes the conclusion SwAcceptableLevD: **Software performs its intended function at acceptable level of safety for Level D**. Recall that the stated general purpose of DO-178C is to provide "guidelines 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." Thus, the argument's conclusion represents a concise statement of the stated purpose of the guidance as applied specifically to Level D software.

[6] The elided text is "subject to the approval of the certification authority." The existence of these words in the quoted sentence is entirely superfluous because those words are implicitly part of *everything* in the guidance.

[7] The size of some arguments, combined with suboptimal handling of the interaction of figures and text in Microsoft® Word®, causes several instances of more-than-desired whitespace throughout the exposition of the arguments.

The four context elements (IntFun, DefAccSafetyFromRegs, GlossaryApplies, and LevelDDef) attached to the concise conclusion provide necessary additional information about its precise meaning. IntFun indicates that the software mentioned in the conclusion is fully described in an external **Description of the intended function of the software**. DefAccSafetyFromRegs indicates that the definition of acceptable level of safety is found in the applicable airworthiness regulations. GlossaryApplies indicates that any words or phrases used in the argument with entries in the DO-178C glossary are to be assumed to have the meaning specified therein. Finally, LevelDSoftware specifies that the meaning of Level D software in the conclusion matches the meaning specified in the guidance.

The argument shows that the DO-178C guidance implicitly establishes the conclusion through two premises (HLRSatSRRefLevD and EOCSatHLRefLevD) and the warrant ArgByCorrectness. Both premises are needed to show that the software correctly performs its intended function. Testing to allocated system requirements might be inadequate to show that software adequately addresses software contributions to system hazards. In situations for which high-level requirements are not a satisfactory refinement of allocated system requirements, software might perfectly satisfy the high-level requirements, yet fail to perform its intended function. The use of the module notation for these premises indicates that they are justified by supporting arguments.

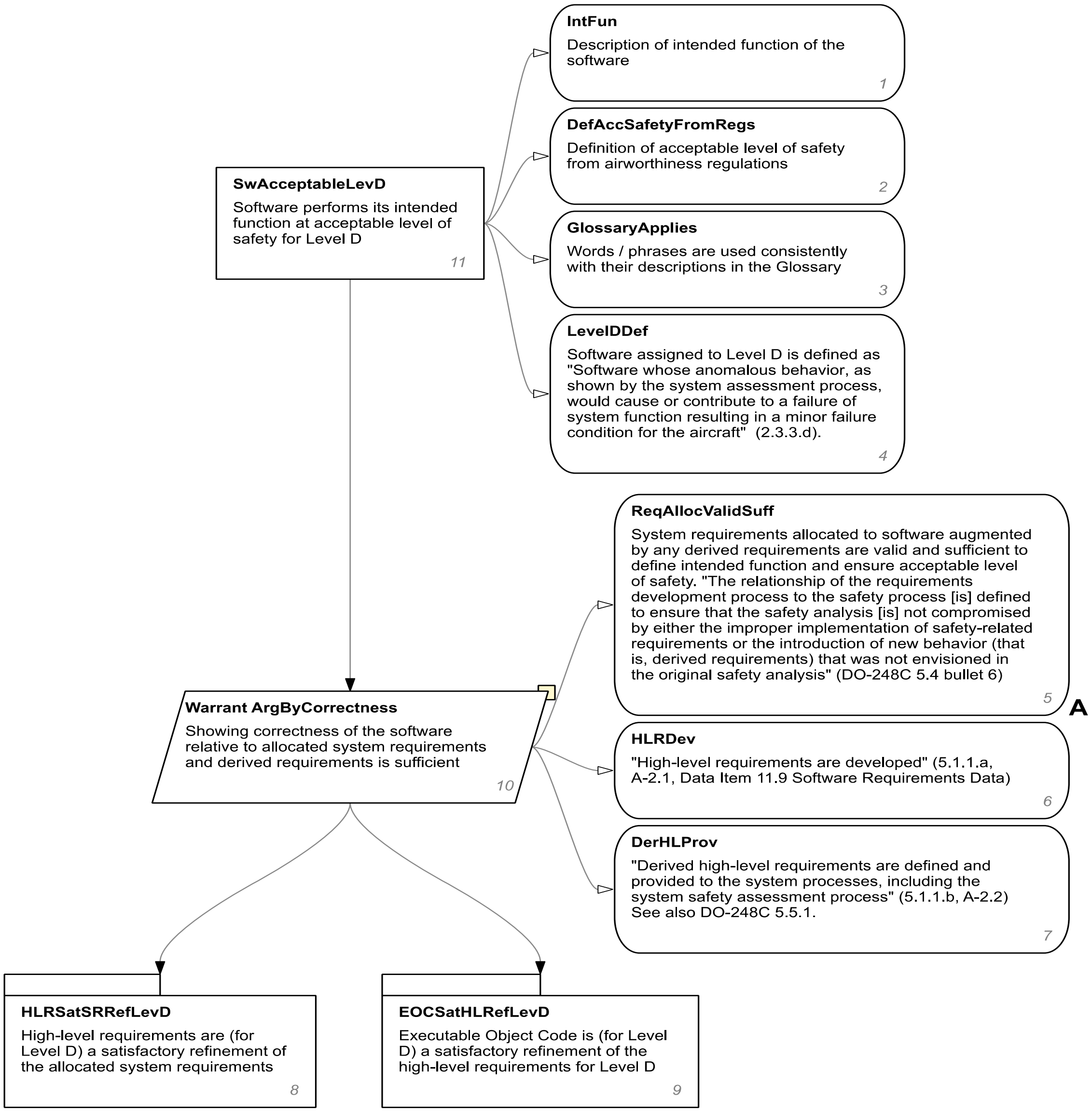

IntFun
Description of intended function of the software
1
DefAccSafetyFromRegs
Definition of acceptable level of safety from airworthiness regulations
2
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
3
LevelDDef
Software assigned to Level D is defined as "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" (2.3.3.d).
4
SwAcceptableLevD
Software performs its intended function at acceptable level of safety for Level D
11
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. "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 original safety analysis" (DO-248C 5.4 bullet 6)
5
A
Warrant ArgByCorrectness
Showing correctness of the software relative to allocated system requirements and derived requirements is sufficient
10
HLRDev
"High-level requirements are developed" (5.1.1.a, A-2.1, Data Item 11.9 Software Requirements Data)
6
DerHLProv
"Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process" (5.1.1.b, A-2.2) See also DO-248C 5.5.1.
7
HLRSatSRRefLevD
High-level requirements are (for Level D) a satisfactory refinement of the allocated system requirements
8
EOCSatHLRefLevD
Executable Object Code is (for Level D) a satisfactory refinement of the high-level requirements for Level D
9

**Figure 3. Level D: SwAcceptableLevD**

The warrant ArgByCorrectness explains that these premises are also sufficient for that purpose. Assumption ReqAllocValidStuff and context elements HLRDev and DerHLProv document how the argument's logic depends on the guidance being used in the context of an effective, compatible system safety assessment process. Without this clarification, the argument might: 1) leave readers wondering how evidence of requirements refinement and satisfaction shows achievement of the claimed level of safety; 2) mislead readers into strictly equating "safety" and software correctness; or 3) give readers the impression that the standard deems such safety analysis unnecessary for Level D software. By documenting both the assumption of an external system safety assessment process and objective A-2.2's requirement that software developers provide system safety assessors with derived requirements, this part of the argument explains how system development efforts using DO-178C address safety despite the explicit omission of safety analysis evidence.

The only remaining element in the diagram is the confidence link (such links are denoted by a partially hidden small yellow square) attached to ArgByCorrectness. The confidence argument indicated by this link establishes the conclusion JUSTIFIEDCONFIDENCELEVD.

Next is the first premise of the top-level argument, which is HLRSATSRREFLEVD: **High-level requirements are (for Level D) a satisfactory refinement of the allocated system requirements.** The (sub-) argument establishing this premise is shown in figure 4.

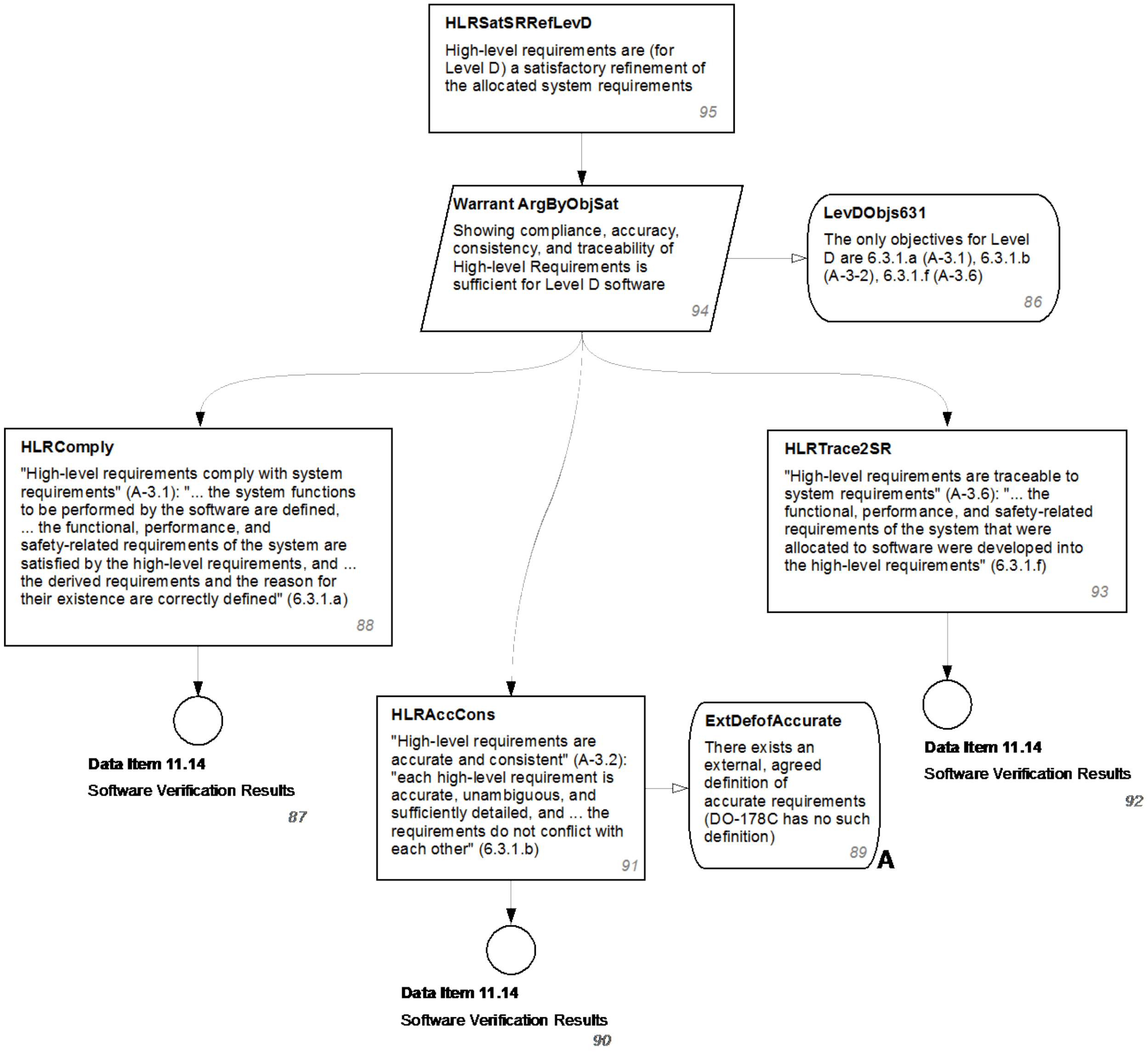

HLRSatSRRefLevD
High-level requirements are (for Level D) a satisfactory refinement of the allocated system requirements
95
Warrant ArgByObjSat
Showing compliance, accuracy, consistency, and traceability of High-level Requirements is sufficient for Level D software
94
LevDObjs631
The only objectives for Level D are 6.3.1.a (A-3.1), 6.3.1.b (A-3-2), 6.3.1.f (A-3.6)
86
HLRComply
"High-level requirements comply with system requirements" (A-3.1): "... the system functions to be performed by the software are defined, ... the functional, performance, and safety-related requirements of the system are satisfied by the high-level requirements, and ... the derived requirements and the reason for their existence are correctly defined" (6.3.1.a)
88
HLRTrace2SR
"High-level requirements are traceable to system requirements" (A-3.6): "... the functional, performance, and safety-related requirements of the system that were allocated to software were developed into the high-level requirements" (6.3.1.f)
93
Data Item 11.14
Software Verification Results
87
HLRAccCons
"High-level requirements are accurate and consistent" (A-3.2): "each high-level requirement is accurate, unambiguous, and sufficiently detailed, and ... the requirements do not conflict with each other" (6.3.1.b)
91
ExtDefofAccurate
There exists an external, agreed definition of accurate requirements (DO-178C has no such definition)
89
A
Data Item 11.14
Software Verification Results
92
Data Item 11.14
Software Verification Results
90

**Figure 4. Level D: HLRSATSRREFLEVD**

The three premises for this conclusion are HLRCOMPLY (which corresponds to the objective stated fully in section 6.3.1.a and summarized in Annex table entry A-3.1), HLRACCCONS (section 6.3.1.b, table entry A-3.2), and HLRTRACE2SR (section 6.3.1.f, table entry A-3.6). HLRCOMPLY concerns the high-level requirements satisfying the system requirements and any derived requirements being handled appropriately. HLRTrace2SR concerns traceability between the high-level and the system requirements allocated to software. HLRAccCons concerns the accuracy, consistency, and unambiguousness of the high-level requirements. Because DO-178C does not provide a definition or description of what is meant by "accuracy," and because the process of applying a standard dictionary definition of the word to requirements is not clear, the assumption is attached (ExtDefofAccurate) that **there exists an external, agreed definition of accurate requirements**. Without such an agreed definition, the meaning of this premise cannot be fully known.

Neither the guidance nor the supporting information provides any explanation for why satisfying these three premises is sufficient (in a Level D sense) to establish that the high-level requirements are a satisfactory refinement of the system requirements allocated to software. Thus, the warrant ArgByObjSat is nothing more than the trivial statement **Showing compliance, accuracy, consistency, and traceability of High-level Requirements is sufficient for Level D software.**

Each of the three premises is supported by direct evidence (which, as noted earlier, is indicated by the small circle) contained in Data Item 11.14: Software Verification Results. This section number and data item name is taken directly from chapter 11 of the guidance.

Figure 5 shows the argument for the second premise (EOCSatHLRefLevD) of the top-level argument. This argument is slightly more complicated than the previous one, involving five premises and several context items that require a bit of explanation. Like the previous argument, however, the warrant ArgByObjSat is a trivial statement because neither the guidance nor supporting information explain why the premises should be considered sufficient to establish the conclusion.

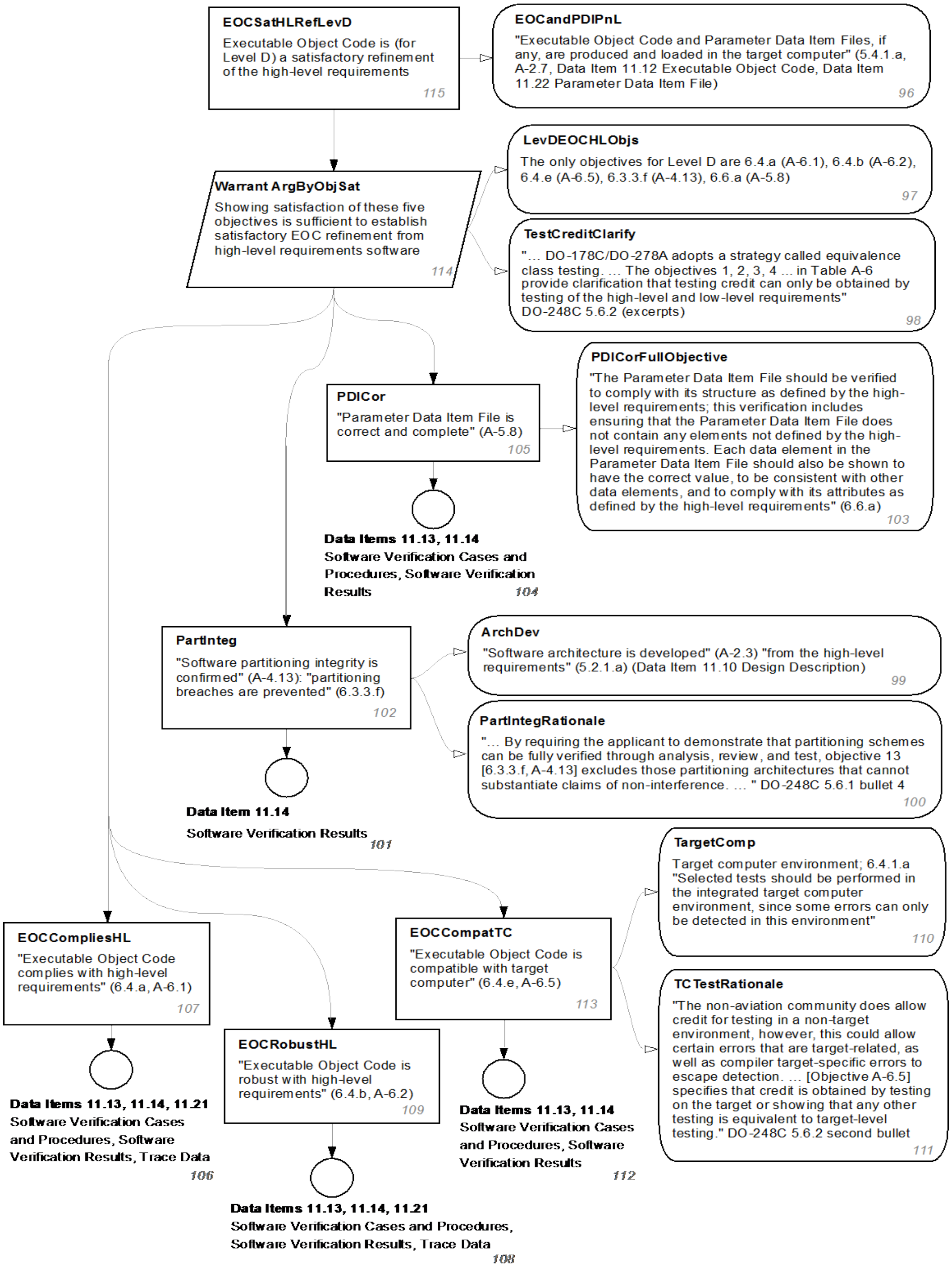

EOCSatHLRefLevD
Executable Object Code is (for Level D) a satisfactory refinement of the high-level requirements
115
EOCandPDIPnL
"Executable Object Code and Parameter Data Item Files, if any, are produced and loaded in the target computer" (5.4.1.a, A-2.7, Data Item 11.12 Executable Object Code, Data Item 11.22 Parameter Data Item File)
96
LevDEOCHLObjs
The only objectives for Level D are 6.4.a (A-6.1), 6.4.b (A-6.2), 6.4.e (A-6.5), 6.3.3.f (A-4.13), 6.6.a (A-5.8)
97
Warrant ArgByObjSat
Showing satisfaction of these five objectives is sufficient to establish satisfactory EOC refinement from high-level requirements software
114
TestCreditClarify
"... DO-178C/DO-278A adopts a strategy called equivalence class testing. ... The objectives 1, 2, 3, 4 ... in Table A-6 provide clarification that testing credit can only be obtained by testing of the high-level and low-level requirements" DO-248C 5.6.2 (excerpts)
98
PDICor
"Parameter Data Item File is correct and complete" (A-5.8)
105
PDICorFullObjective
"The Parameter Data Item File should be verified to comply with its structure as defined by the high-level requirements; this verification includes ensuring that the Parameter Data Item File does not contain any elements not defined by the high-level requirements. Each data element in the Parameter Data Item File should also be shown to have the correct value, to be consistent with other data elements, and to comply with its attributes as defined by the high-level requirements" (6.6.a)
103
Data Items 11.13, 11.14
Software Verification Cases and Procedures, Software Verification Results
104
PartInteg
"Software partitioning integrity is confirmed" (A-4.13): "partitioning breaches are prevented" (6.3.3.f)
102
ArchDev
"Software architecture is developed" (A-2.3) "from the high-level requirements" (5.2.1.a) (Data Item 11.10 Design Description)
99
PartIntegRationale
"... By requiring the applicant to demonstrate that partitioning schemes can be fully verified through analysis, review, and test, objective 13 [6.3.3.f, A-4.13] excludes those partitioning architectures that cannot substantiate claims of non-interference. ... " DO-248C 5.6.1 bullet 4
100
Data Item 11.14
Software Verification Results
101
TargetComp
Target computer environment; 6.4.1.a "Selected tests should be performed in the integrated target computer environment, since some errors can only be detected in this environment"
110
EOCCompliesHL
"Executable Object Code complies with high-level requirements" (6.4.a, A-6.1)
107
EOCCompatTC
"Executable Object Code is compatible with target computer" (6.4.e, A-6.5)
113
TC TestRationale
"The non-aviation community does allow credit for testing in a non-target environment, however, this could allow certain errors that are target-related, as well as compiler target-specific errors to escape detection. ... [Objective A-6.5] specifies that credit is obtained by testing on the target or showing that any other testing is equivalent to target-level testing." DO-248C 5.6.2 second bullet
111
EOCRobustHL
"Executable Object Code is robust with high-level requirements" (6.4.b, A-6.2)
109
Data Items 11.13, 11.14, 11.21
Software Verification Cases and Procedures, Software Verification Results, Trace Data
106
Data Items 11.13, 11.14
Software Verification Cases and Procedures, Software Verification Results
112
Data Items 11.13, 11.14, 11.21
Software Verification Cases and Procedures, Software Verification Results, Trace Data
108

**Figure 5. Level D: EOCSatHLRefLevD**

The context item attached to the conclusion requires explanation. EOCandPDIPnL encapsulates the objective of the integration process stated in section 5.4.1 and table A-2.7 of DO-178C: the **“Executable Object Code and Parameter Data Item Files, if any, are produced and loaded in the target computer”** [1] An objective is captured as a context element in the argument instead of as a conclusion or premise because any objective concerning only the existence of a data item is consistently represented as context elements. This is because showing the satisfaction of such an objective does not require any additional argument.

Of the five premises, three directly address the quality of the executable object code. The evidence for two of these (EOCCompliesHL, EOCRobustHL) is contained in the relevant parts of three data items:

- Data Item 11.13: Software Verification Cases and Procedures
- Data Item 11.14: Software Verification Results
- Data Item 11.21: Trace Data

The third premise concerning the executable object code (EOCCompatTC) concerns compatibility with the target computer. Two context elements (TargetComp, TCTestRationale) provide additional information regarding why testing on the target computer is considered essential. The evidence for demonstrating the truth of this premise is contained in Data Item 11.13 and Data Item 11.14.

The other two premises are not directly related to executable object code. PartInteg concerns the integrity of software partitioning. The context element PartIntegRationale explains the reason for requiring a showing of partitioning integrity. Context ArchDev indicates that developing the software architecture is needed to define what partitioning is employed.

The final premise in this argument is PDICor: **“Parameter Data Item File is correct and complete” (A-5.8)**. The text is the simplified version of the full objective given in the annex table. The full text of the objective is written in the attached context element to enhance the look of the diagram.

The associated confidence argument is shown in figure 6.

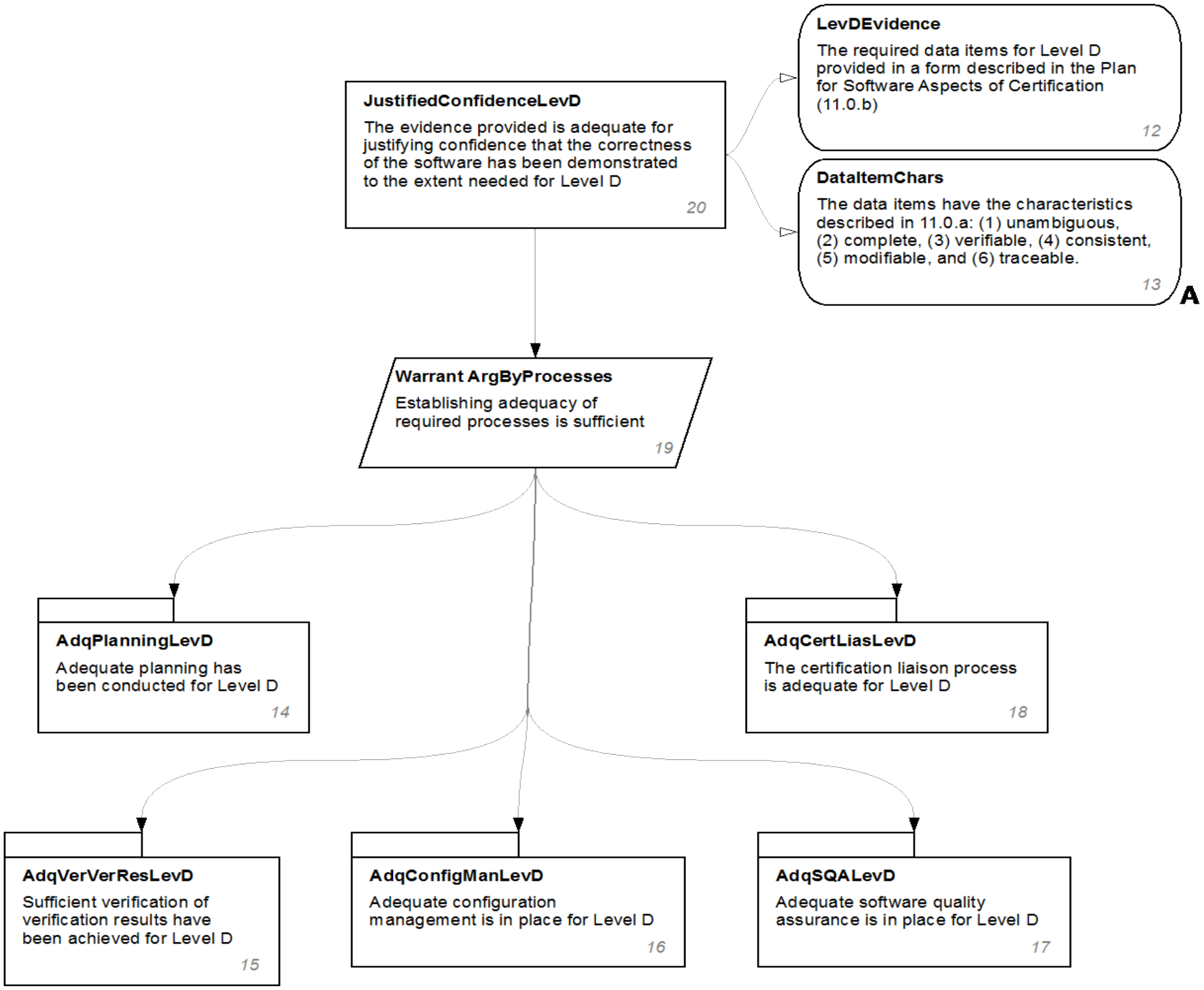

LevDEvidence
The required data items for Level D provided in a form described in the Plan for Software Aspects of Certification (11.0.b)
12
JustifiedConfidenceLevD
The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for Level D
20
DataItemChars
The data items have the characteristics described in 11.0.a: (1) unambiguous, (2) complete, (3) verifiable, (4) consistent, (5) modifiable, and (6) traceable.
13
A
Warrant ArgByProcesses
Establishing adequacy of required processes is sufficient
19
AdqPlanningLevD
Adequate planning has been conducted for Level D
14
AdqCertLiasLevD
The certification liaison process is adequate for Level D
18
AdqVerVerResLevD
Sufficient verification of verification results have been achieved for Level D
15
AdqConfigManLevD
Adequate configuration management is in place for Level D
16
AdqSQALevD
Adequate software quality assurance is in place for Level D
17

**Figure 6. Level D: JustifiedConfidenceLevD**

The conclusion of the confidence argument is JUSTIFIEDCONFIDENCELEVD: **The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for Level D**. This argument encapsulates all of the Level D relevant objectives from DO-178C that concern required processes. It asserts explicitly, as the guidance asserts implicitly, that establishing adequacy of required processes is sufficient to provide the needed confidence. The adequacy of these required processes is established in five premises, which are expanded in supporting arguments: ADQPLANNINGLEVD, ADQVERVERLEVD, ADQCONFIGMANLEVD, ADQSQALEVD, ADQCERTLIASLEVD.

Associated with the conclusion is a context element (LEVDEVIDENCE) and an assumption (DATAITEMCHARS). The context element notes that the data items are to be provided in a form described in the Plan for Software Aspects of Certification (11.0.b). The assumption lists the characteristics (described in 11.0.a) that each data item is supposed to possess. It should be: **1) unambiguous; 2) complete; 3) verifiable; 4) consistent; 5) modifiable; and 6) traceable**. These characteristics are enumerated in an assumption, rather than as a conclusion of a supporting argument because DO-178C does not contain any objectives requiring that the characteristics be directly demonstrated.

Figure 7 contains the simple argument justifying AdqPlanningLevD.

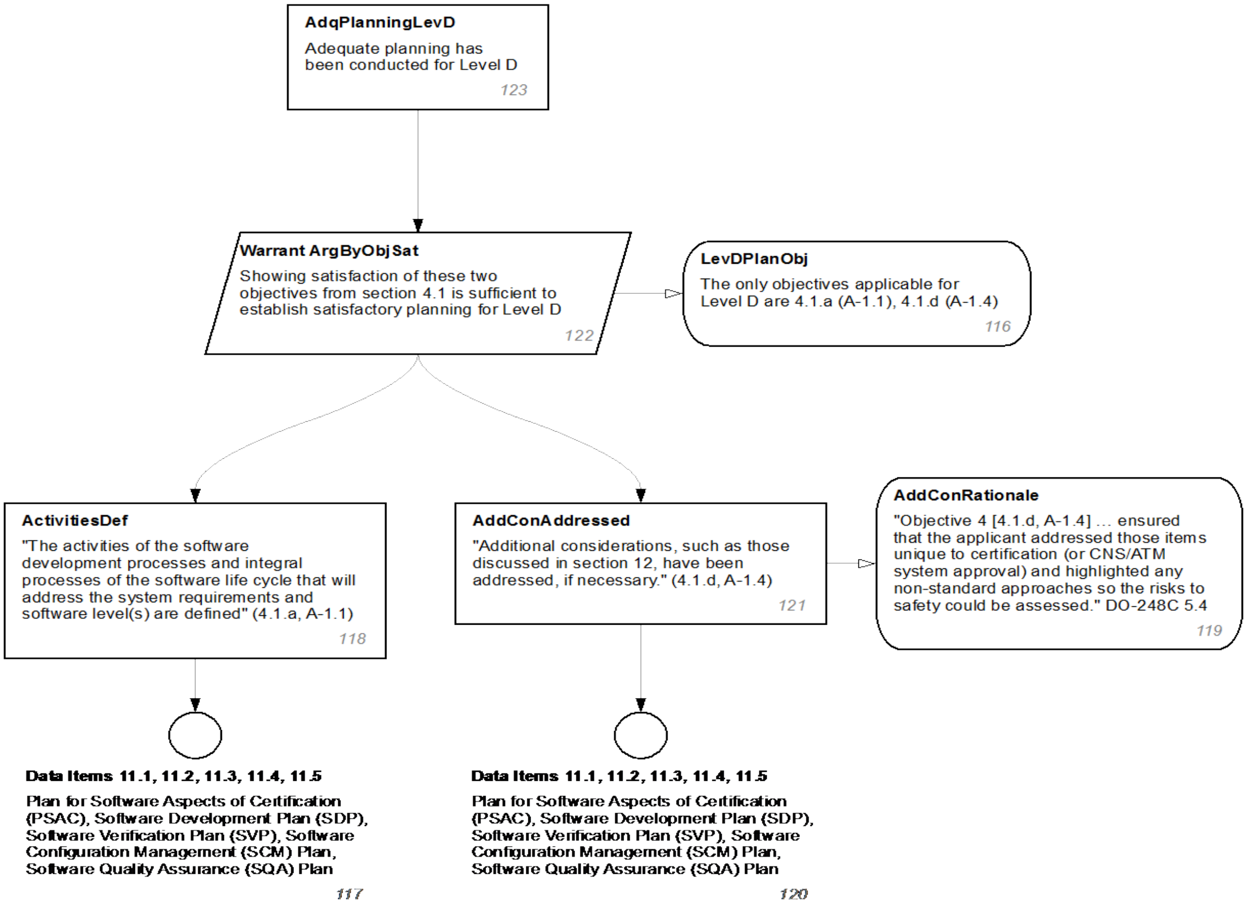

AdqPlanningLevD
Adequate planning has been conducted for Level D
123
Warrant ArgByObjSat
Showing satisfaction of these two objectives from section 4.1 is sufficient to establish satisfactory planning for Level D
122
LevDPlanObj
The only objectives applicable for Level D are 4.1.a (A-1.1), 4.1.d (A-1.4)
116
ActivitiesDef
"The activities of the software development processes and integral processes of the software life cycle that will address the system requirements and software level(s) are defined" (4.1.a, A-1.1)
118
AddConAddressed
"Additional considerations, such as those discussed in section 12, have been addressed, if necessary." (4.1.d, A-1.4)
121
AddConRationale
"Objective 4 [4.1.d, A-1.4] ... ensured that the applicant addressed those items unique to certification (or CNS/ATM system approval) and highlighted any non-standard approaches so the risks to safety could be assessed." DO-248C 5.4
119
Data Items 11.1, 11.2, 11.3, 11.4, 11.5
Plan for Software Aspects of Certification (PSAC), Software Development Plan (SDP), Software Verification Plan (SVP), Software Configuration Management (SCM) Plan, Software Quality Assurance (SQA) Plan
117
Data Items 11.1, 11.2, 11.3, 11.4, 11.5
Plan for Software Aspects of Certification (PSAC), Software Development Plan (SDP), Software Verification Plan (SVP), Software Configuration Management (SCM) Plan, Software Quality Assurance (SQA) Plan
120

**Figure 7. Level D: AdqPlanningLevD**

At Level D, the guidance requires only two objectives related to planning. Satisfying these two objectives is explicated in the two premises: ActivitiesDef and AddConAddressed. The evidence required to show these two premises hold is contained in the relevant parts of five data items:

- Data Item 11.1: Plan for Software Aspects of Certification (PSAC)
- Data Item 11.2: Software Development Plan (SDP)
- Data Item 11.3: Software Verification Plan (SVP)
- Data Item 11.4: Software Configuration Management (SCM) Plan
- Data Item 11.5: Software Quality Assurance (SQA) Plan

Because neither the guidance nor supporting information explains why these two premises are considered sufficient, the warrant takes the same trivial form previously seen.

Figure 8 shows the simple argument describing the requirements for Level D for the verification of the results of the verification process, which is given the slightly shorter name **verification of verification results** in the argument. No additional commentary seems necessary.

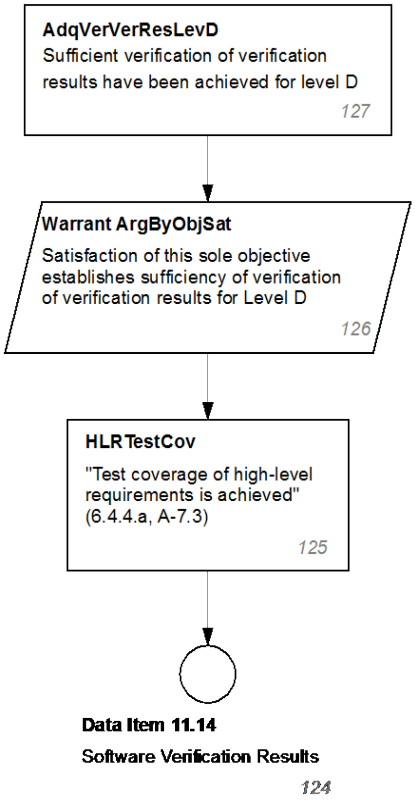

AdqVerVerResLevD
Sufficient verification of verification results have been achieved for level D
127
Warrant ArgByObjSat
Satisfaction of this sole objective establishes sufficiency of verification of verification results for Level D
126
HLRTestCov
"Test coverage of high-level requirements is achieved" (6.4.4.a, A-7.3)
125
Data Item 11.14
Software Verification Results
124

**Figure 8. Level D: ADQVERVERLEVD**

In contract, the argument concerning configuration management is a bit more complicated, as shown in figure 9.

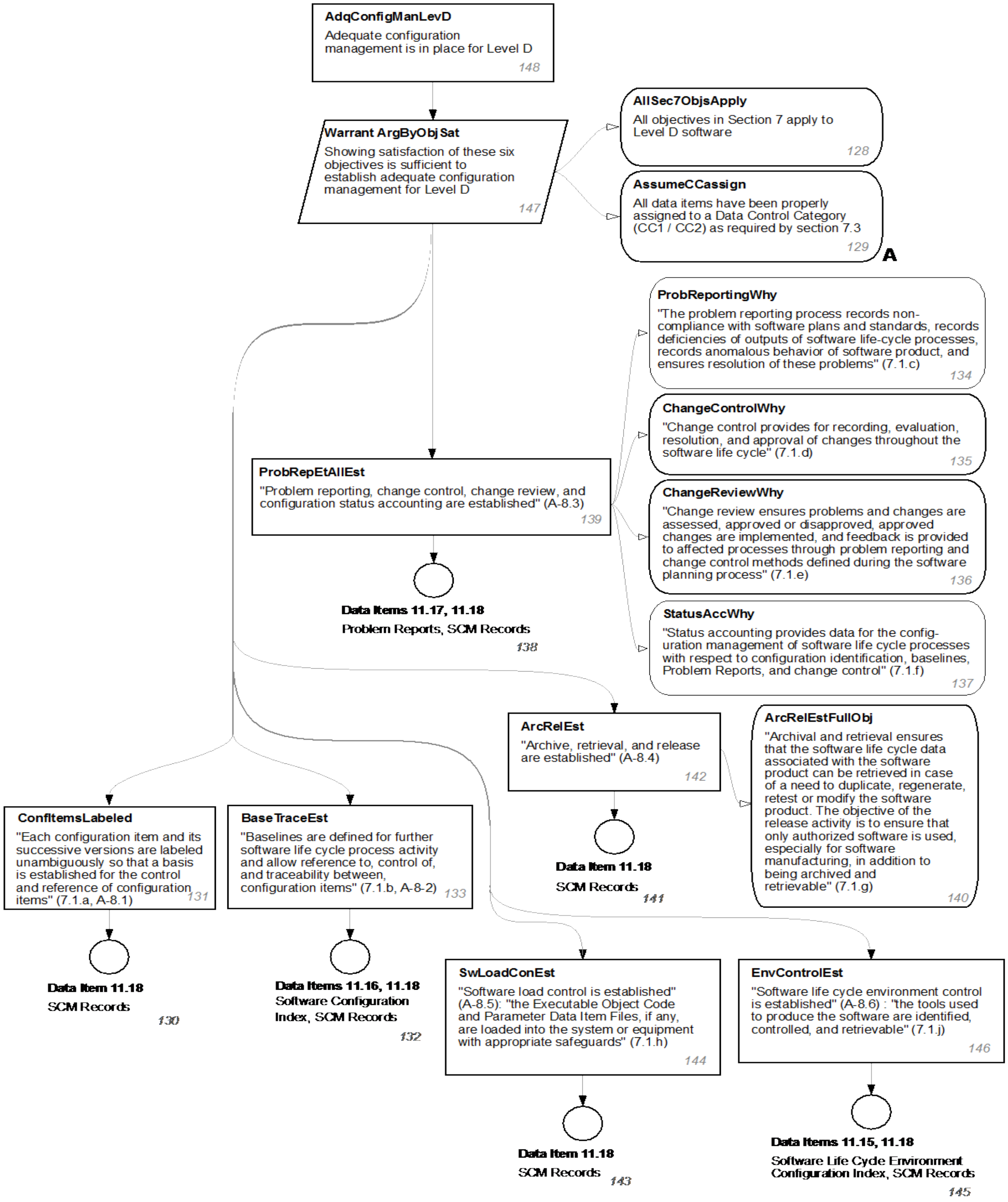

AdqConfigManLevD
Adequate configuration management is in place for Level D
148
Warrant ArgByObjSat
Showing satisfaction of these six objectives is sufficient to establish adequate configuration management for Level D
147
AllSec7ObjsApply
All objectives in Section 7 apply to Level D software
128
AssumeCCassign
All data items have been properly assigned to a Data Control Category (CC1 / CC2) as required by section 7.3
129
A
ProbReportingWhy
"The problem reporting process records non-compliance with software plans and standards, records deficiencies of outputs of software life-cycle processes, records anomalous behavior of software product, and ensures resolution of these problems" (7.1.c)
134
ChangeControlWhy
"Change control provides for recording, evaluation, resolution, and approval of changes throughout the software life cycle" (7.1.d)
135
ChangeReviewWhy
"Change review ensures problems and changes are assessed, approved or disapproved, approved changes are implemented, and feedback is provided to affected processes through problem reporting and change control methods defined during the software planning process" (7.1.e)
136
StatusAccWhy
"Status accounting provides data for the configuration management of software life cycle processes with respect to configuration identification, baselines, Problem Reports, and change control" (7.1.f)
137
ProbRepEtAllEst
"Problem reporting, change control, change review, and configuration status accounting are established" (A-8.3)
139
Data Items 11.17, 11.18
Problem Reports, SCM Records
138
ArcRelEst
"Archive, retrieval, and release are established" (A-8.4)
142
ArcRelEstFullObj
"Archival and retrieval ensures that the software life cycle data associated with the software product can be retrieved in case of a need to duplicate, regenerate, retest or modify the software product. The objective of the release activity is to ensure that only authorized software is used, especially for software manufacturing, in addition to being archived and retrievable" (7.1.g)
140
Data Item 11.18
SCM Records
141
ConfItemsLabeled
"Each configuration item and its successive versions are labeled unambiguously so that a basis is established for the control and reference of configuration items" (7.1.a, A-8.1)
131
BaseTraceEst
"Baselines are defined for further software life cycle process activity and allow reference to, control of, and traceability between, configuration items" (7.1.b, A-8-2)
133
Data Item 11.18
SCM Records
130
Data Items 11.16, 11.18
Software Configuration Index, SCM Records
132
SwLoadConEst
"Software load control is established" (A-8.5): "the Executable Object Code and Parameter Data Item Files, if any, are loaded into the system or equipment with appropriate safeguards" (7.1.h)
144
EnvControlEst
"Software life cycle environment control is established" (A-8.6) : "the tools used to produce the software are identified, controlled, and retrievable" (7.1.j)
146
Data Item 11.18
SCM Records
143
Data Items 11.15, 11.18
Software Life Cycle Environment Configuration Index, SCM Records
145

**Figure 9. Level D: ADQCONFIGMANLEVD**

The conclusion ADQCONFIGMANLEVD is supported by six premises ConfItemsLabeled, BaseTraceEst, ProbRepEtAllEst, ArcRelEst, SwLoadConEst, and EnvControlEst, which identify necessary source control properties and capabilities. According to the guidance, unless each of the versions of each configuration item is unambiguously labeled, the relationship between the argument text and these artifacts would be unclear, and the validity of evidence would be in doubt. Without baselines, traceability between the various items and the items that informed their production could not be established. Without problem reporting, problems might be identified only to fall through the cracks. Without change control and change review, it would be necessary to follow a strict waterfall process, re-executing each stage in its entirety whenever a discovered defect forces a change. Configuration status accounting information underpins the other configuration management activities. Archive, retrieval, and release capacities make it possible to revisit any past version as needed to investigate or address problems. Software load control ensures that only authorized versions of the executable object code and parameter data items are used in airborne systems and that it is known which versions are in use in which systems. Without software life cycle environment control, it would not be possible to precisely recreate given versions of artifacts, such as compiled executable object code, from their source artifacts (e.g., as different versions of a compiler might produce slightly different code).

Context elements, such as ProbReportingWhy and ArcRelEstFullObj, clarify what the argument means when it names configuration management practices and capabilities. For example, ProbReportingWhy explains what problem reporting is by describing how it operates and what it accomplishes. ArcRelEstFullObj explains **archive, retrieval, and release** by observing that this capability permits developers to retrieve software life-cycle data **in case of a need to duplicate, regenerate, retest, or modify the software product.**

Warrant ArgByObjSat asserts that the identified practices and capabilities are sufficient. This sufficiency is not justified by explicit backing evidence because the rationale is not stated.

Context element AllSec7ObjsApply shows which of DO-178C’s objectives relate to configuration management and that all such objectives apply at Level D and all higher software levels. Assumption AssumeCCassign serves multiple purposes. First, it clarifies that the configuration items discussed in this argument includes all data items named in the standard. Second, it identifies the leveling mechanism at work in configuration management. Whereas all objectives apply to at least some configuration items at every software level, some objectives do not apply to some configuration items at some levels. The tables in Annex A of the section specify the control category of each data item at each software level, whereas Section 7.3 of the standard defines which objectives apply at which control category.

The meanings of the cited evidence items should be easily discernable based on analogy to the explanations given for previous evidence items. Going forward, no mention will be made of evidence items in the explanatory text.

Figure 10 is related to a simple argument, which relies on two premises to justify the conclusion based ADQSQALEVD: **Adequate software quality assurance is in place for Level D**.

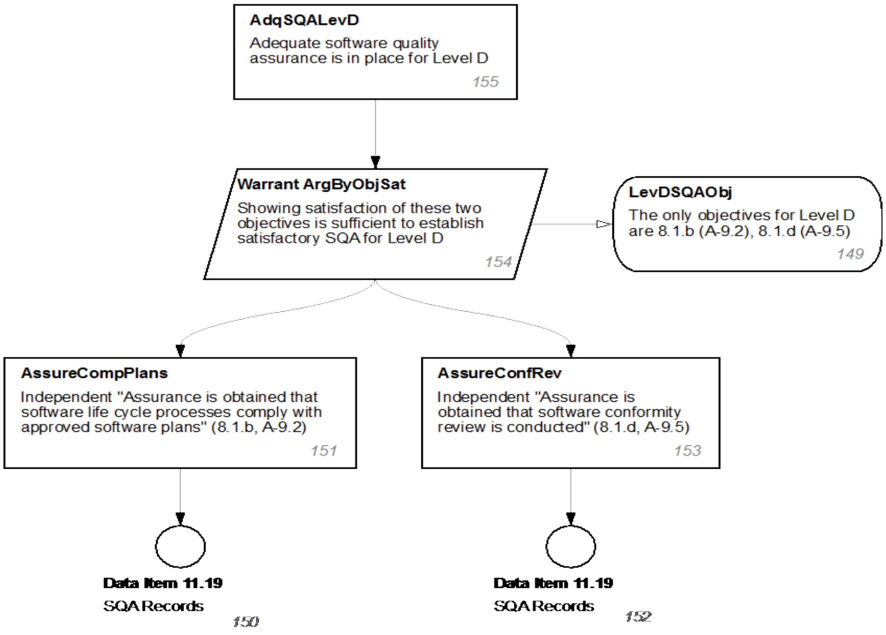

AdqSQALevD
Adequate software quality assurance is in place for Level D
155
Warrant ArgByObjSat
Showing satisfaction of these two objectives is sufficient to establish satisfactory SQA for Level D
154
LevDSQAObj
The only objectives for Level D are 8.1.b (A-9.2), 8.1.d (A-9.5)
149
AssureCompPlans
Independent "Assurance is obtained that software life cycle processes comply with approved software plans" (8.1.b, A-9.2)
151
AssureConfRev
Independent "Assurance is obtained that software conformity review is conducted" (8.1.d, A-9.5)
153
Data Item 11.19
SQA Records
150
Data Item 11.19
SQA Records
152

**Figure 10. Level D: AdqSQALevD**

Figure 11 shows a simple, three-premise argument concerning the certification liaison process. Interestingly, the objectives required by DO-178C for this process are the same for software Levels D through A. Thus, the confidence arguments for Levels C, B, and A all reference directly the supporting argument presented here for Level D.

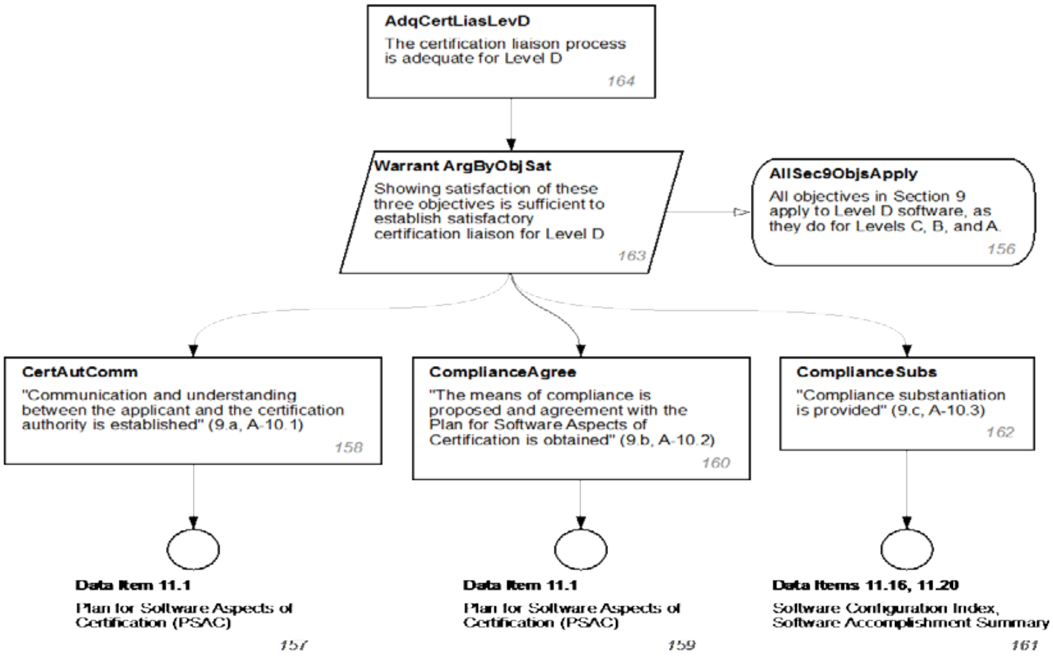

AdqCertLiasLevD
The certification liaison process is adequate for Level D
164
Warrant ArgByObjSat
Showing satisfaction of these three objectives is sufficient to establish satisfactory certification liaison for Level D
163
AllSec9ObjsApply
All objectives in Section 9 apply to Level D software, as they do for Levels C, B, and A.
156
CertAutComm
"Communication and understanding between the applicant and the certification authority is established" (9.a, A-10.1)
158
ComplianceAgree
"The means of compliance is proposed and agreement with the Plan for Software Aspects of Certification is obtained" (9.b, A-10.2)
160
ComplianceSubs
"Compliance substantiation is provided" (9.c, A-10.3)
162
Data Item 11.1
Plan for Software Aspects of Certification (PSAC)
157
Data Item 11.1
Plan for Software Aspects of Certification (PSAC)
159
Data Items 11.16, 11.20
Software Configuration Index, Software Accomplishment Summary
161

**Figure 11. Level D: AdqCertLiasLevD**

## 3.3 LEVEL C ARGUMENTS

Although section 3.2 presented the full set of Level D arguments, only a subset of the arguments associated with Level C is presented here. Omitted arguments are shown in appendix A. Explanatory text is provided only for aspects of the arguments that bring out ideas that have not already been addressed.

Figure 12 contains the main argument for Level C, which establishes the conclusion SWACCEPTABLELEVC: **Software performs its intended function at acceptable level of safety for Level C**. The argument has the same structure as the main Level D argument. The content is directly analogous as well, with the substitutions appropriate for the different software level. All but one of the context elements, the assumption, and the warrant are unchanged. The fourth context element provides the definition for Level C (instead of D) software.

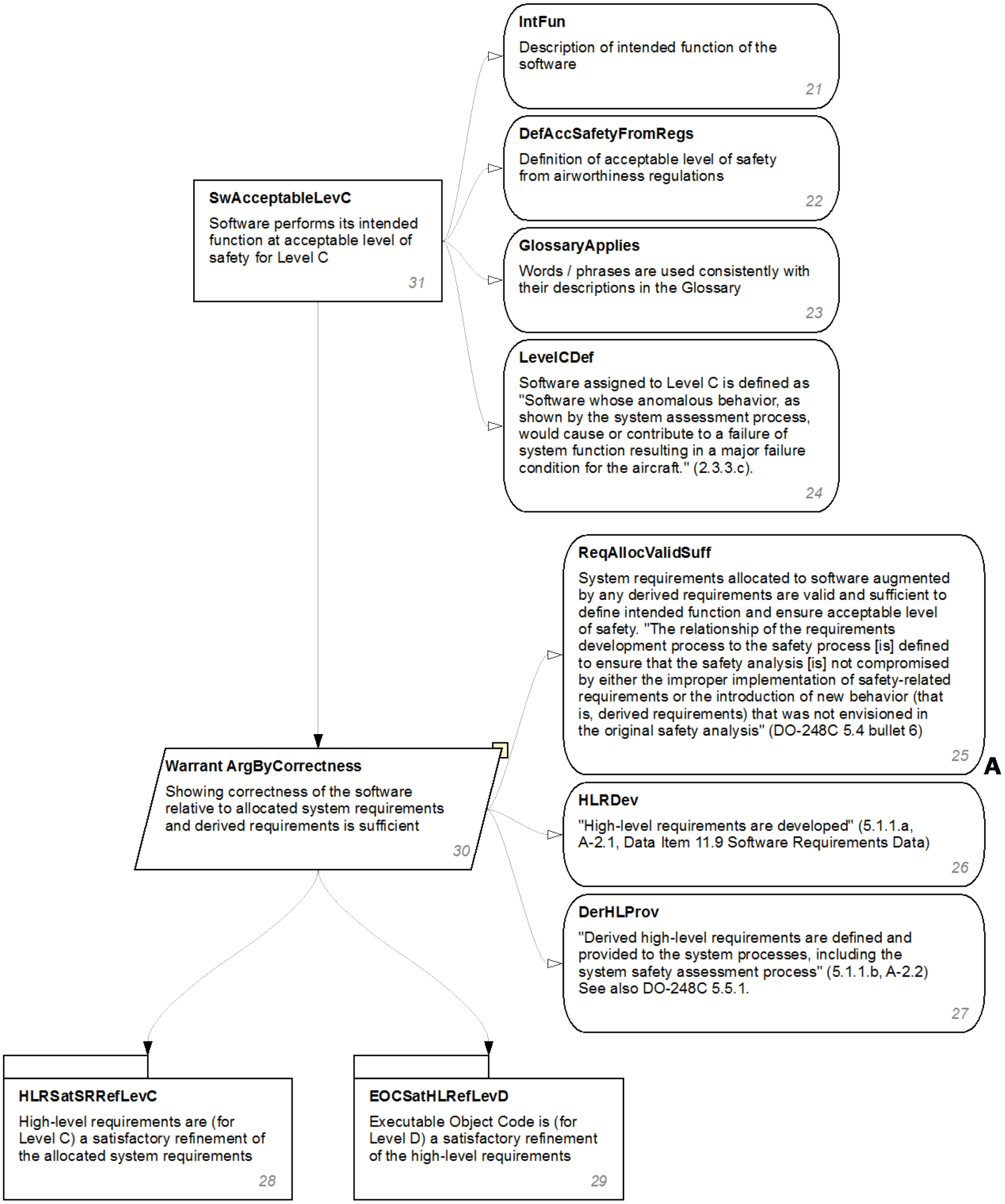

IntFun
Description of intended function of the software
21
DefAccSafetyFromRegs
Definition of acceptable level of safety from airworthiness regulations
22
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
23
LevelCDef
Software assigned to Level C is defined as "Software whose anomalous behavior, as shown by the system assessment process, would cause or contribute to a failure of system function resulting in a major failure condition for the aircraft." (2.3.3.c).
24
SwAcceptableLevC
Software performs its intended function at acceptable level of safety for Level C
31
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. "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 original safety analysis" (DO-248C 5.4 bullet 6)
25
A
Warrant ArgByCorrectness
Showing correctness of the software relative to allocated system requirements and derived requirements is sufficient
30
HLRDev
"High-level requirements are developed" (5.1.1.a, A-2.1, Data Item 11.9 Software Requirements Data)
26
DerHLProv
"Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process" (5.1.1.b, A-2.2) See also DO-248C 5.5.1.
27
HLRSatSRRefLevC
High-level requirements are (for Level C) a satisfactory refinement of the allocated system requirements
28
EOCSatHLRefLevD
Executable Object Code is (for Level D) a satisfactory refinement of the high-level requirements
29

**Figure 12. Level C: SwAcceptableLevC**

One of the premises is unchanged. Table A-6 in [1] shows why the premise is EOCSatHLRefLevD instead of EOCSatHLRefLevC. The table shows that the objectives are the same between Levels C and D concerning the relationship between executable object code and high-level requirements.

The other premise (HLRSatSRRefLevC) is changed and requires a new supporting argument to substantiate it. This argument is shown in figure 13.

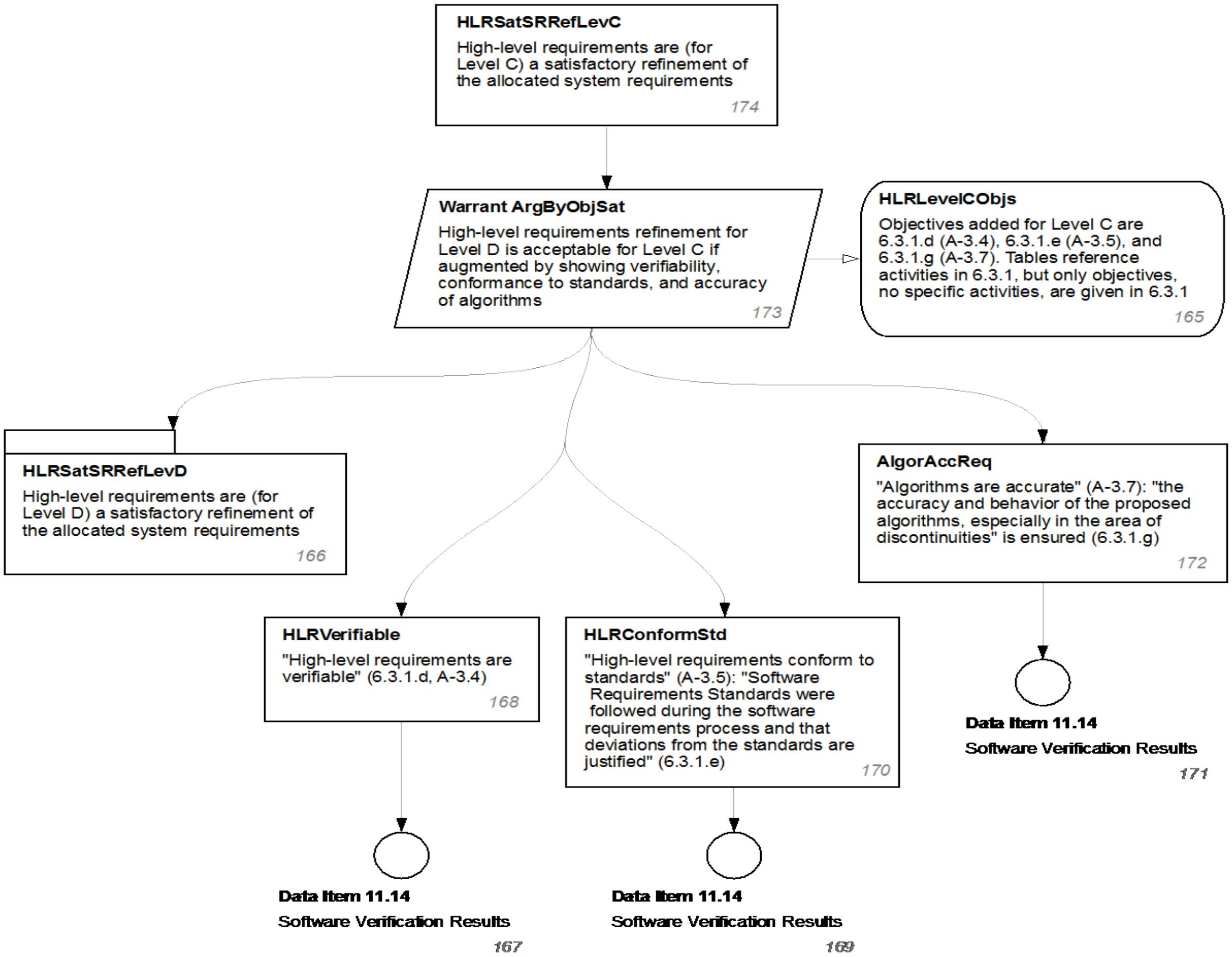

HLRSatSRRefLevC
High-level requirements are (for Level C) a satisfactory refinement of the allocated system requirements
174
Warrant ArgByObjSat
High-level requirements refinement for Level D is acceptable for Level C if augmented by showing verifiability, conformance to standards, and accuracy of algorithms
173
HLRLevelCObjs
Objectives added for Level C are 6.3.1.d (A-3.4), 6.3.1.e (A-3.5), and 6.3.1.g (A-3.7). Tables reference activities in 6.3.1, but only objectives, no specific activities, are given in 6.3.1
165
HLRSatSRRefLevD
High-level requirements are (for Level D) a satisfactory refinement of the allocated system requirements
166
AlgorAccReq
"Algorithms are accurate" (A-3.7): "the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities" is ensured (6.3.1.g)
172
HLRVerifiable
"High-level requirements are verifiable" (6.3.1.d, A-3.4)
168
HLRConformStd
"High-level requirements conform to standards" (A-3.5): "Software Requirements Standards were followed during the software requirements process and that deviations from the standards are justified" (6.3.1.e)
170
Data Item 11.14
Software Verification Results
171
Data Item 11.14
Software Verification Results
167
Data Item 11.14
Software Verification Results
169

**Figure 13. Level C: HLRSatSRRefLevC**

The guidance for Level C introduces objectives about low-level requirements. Therefore, one may wonder why there is nothing about low-level requirements in the main argument. The first version of the case for Level C included a premise about the refinement of low-level requirements. Further reflection suggested that the confidence argument might be a better place to capture low-level requirement objectives. Several DO-178C experts concurred with the opinion; therefore, the final version follows that approach. The results are shown in figures 14 and 17.

According to the DO-178C guidance for Level C, to show satisfactory refinement of allocated system requirements into high-level requirements, it must be demonstrated that HLRSATSRREFLEVD holds, along with three additional premises:

- HLRVERIFIABLE: **"High-level requirements are verifiable" (6.3.1.d, A-3.4)**
- HLRCONFORMSTD: **"High-level requirements conform to standards" (A-3.5): "Software Requirements Standards were followed during the software requirements process and that deviations from the standards are justified" (6.3.1.e)**
- ALGORACCREQ: **"Algorithms are accurate" (A-3.7): "the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities" is ensured (6.3.1.g)**

Each of these is shown to hold by reference to the evidence contained in DATA ITEM 11.14: Software Verification Results. As shown in several previous instances, neither the guidance nor the supporting information provides an explicit rationale for why these additional objectives are necessary or sufficient. Thus, the warrant takes the default, trivial form.

The confidence argument for Level C (figure 14) looks similar to the confidence argument for Level D. One of the premises is identical (ADQCERTLIASLEVD) because no new objectives are added for the certification liaison process. Four of the premises are directly analogous, with new supporting arguments required to address added objectives. One premise (ADDREFINELEVELCSAT) is entirely new. It addresses the additional refinement steps that are introduced in the guidance for Level C.

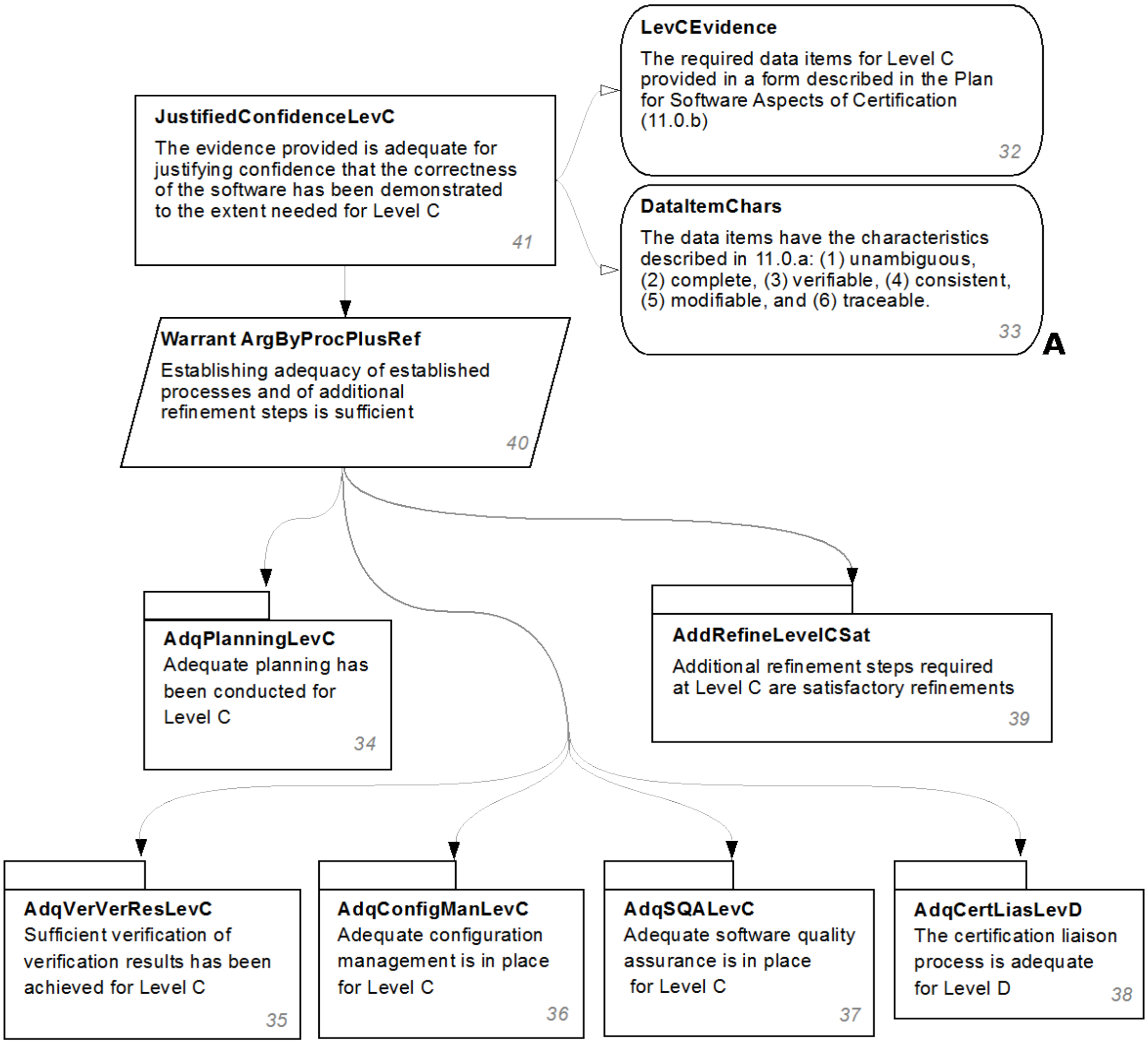

LevCEvidence
The required data items for Level C provided in a form described in the Plan for Software Aspects of Certification (11.0.b)
32
DataItemChars
The data items have the characteristics described in 11.0.a: (1) unambiguous, (2) complete, (3) verifiable, (4) consistent, (5) modifiable, and (6) traceable.
33
A
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
41
Warrant ArgByProcPlusRef
Establishing adequacy of established processes and of additional refinement steps is sufficient
40
AdqPlanningLevC
Adequate planning has been conducted for Level C
34
AddRefineLevelCSat
Additional refinement steps required at Level C are satisfactory refinements
39
AdqVerVerResLevC
Sufficient verification of verification results has been achieved for Level C
35
AdqConfigManLevC
Adequate configuration management is in place for Level C
36
AdqSQALevC
Adequate software quality assurance is in place for Level C
37
AdqCertLiasLevD
The certification liaison process is adequate for Level D
38

**Figure 14. Level C: JustifiedConfidenceLevC**

The argument for adequate planning at Level D contained only two premises. The argument for adequate planning at Level C (see figure 15) includes the Level D conclusion as a premise and an additional five premises. This argument applies to Level B and Level A, also, because no new planning objectives are added at those software levels. One peculiarity exists in the argument, namely existence of the intermediate premise LEVCPLANSAT; which is presented only for explanatory and aesthetic purposes. All premises beneath it could be directly connected to the warrant.

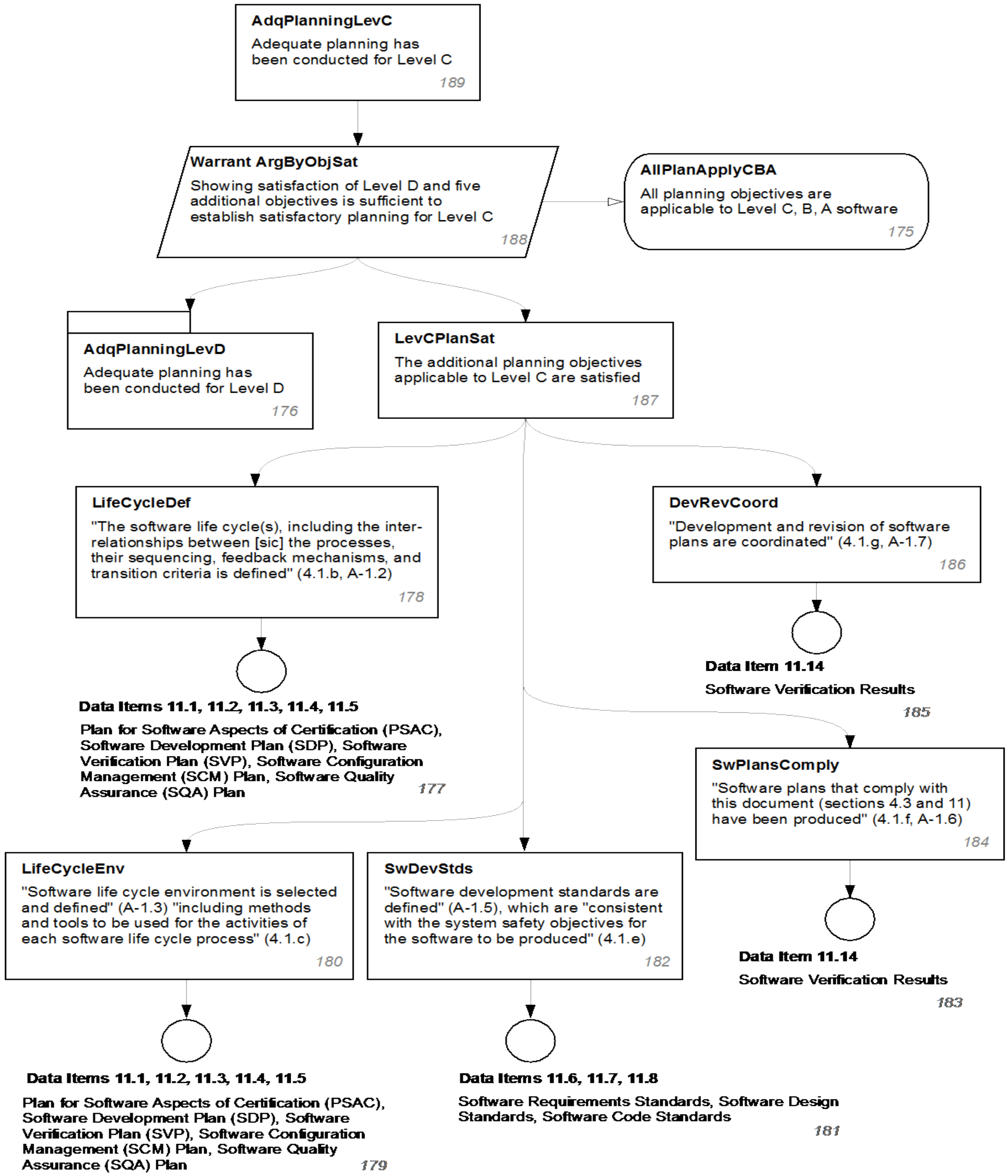

AdqPlanningLevC
Adequate planning has been conducted for Level C
189
Warrant ArgByObjSat
Showing satisfaction of Level D and five additional objectives is sufficient to establish satisfactory planning for Level C
188
AllPlanApplyCBA
All planning objectives are applicable to Level C, B, A software
175
AdqPlanningLevD
Adequate planning has been conducted for Level D
176
LevCPlanSat
The additional planning objectives applicable to Level C are satisfied
187
LifeCycleDef
"The software life cycle(s), including the inter-relationships between [sic] the processes, their sequencing, feedback mechanisms, and transition criteria is defined" (4.1.b, A-1.2)
178
DevRevCoord
"Development and revision of software plans are coordinated" (4.1.g, A-1.7)
186
Data Item 11.14
Software Verification Results
185
Data Items 11.1, 11.2, 11.3, 11.4, 11.5
Plan for Software Aspects of Certification (PSAC), Software Development Plan (SDP), Software Verification Plan (SVP), Software Configuration Management (SCM) Plan, Software Quality Assurance (SQA) Plan
177
SwPlansComply
"Software plans that comply with this document (sections 4.3 and 11) have been produced" (4.1.f, A-1.6)
184
LifeCycleEnv
"Software life cycle environment is selected and defined" (A-1.3) "including methods and tools to be used for the activities of each software life cycle process" (4.1.c)
180
SwDevStds
"Software development standards are defined" (A-1.5), which are "consistent with the system safety objectives for the software to be produced" (4.1.e)
182
Data Item 11.14
Software Verification Results
183
Data Items 11.1, 11.2, 11.3, 11.4, 11.5
Plan for Software Aspects of Certification (PSAC), Software Development Plan (SDP), Software Verification Plan (SVP), Software Configuration Management (SCM) Plan, Software Quality Assurance (SQA) Plan
179
Data Items 11.6, 11.7, 11.8
Software Requirements Standards, Software Design Standards, Software Code Standards
181

**Figure 15. Level C: AdqPlanningLevC**

Figure 16 presents the argument establishing the sufficiency of the verification of verification results. It is similar in form to the planning argument because it includes the Level D conclusion as a premise, along with five new ones.

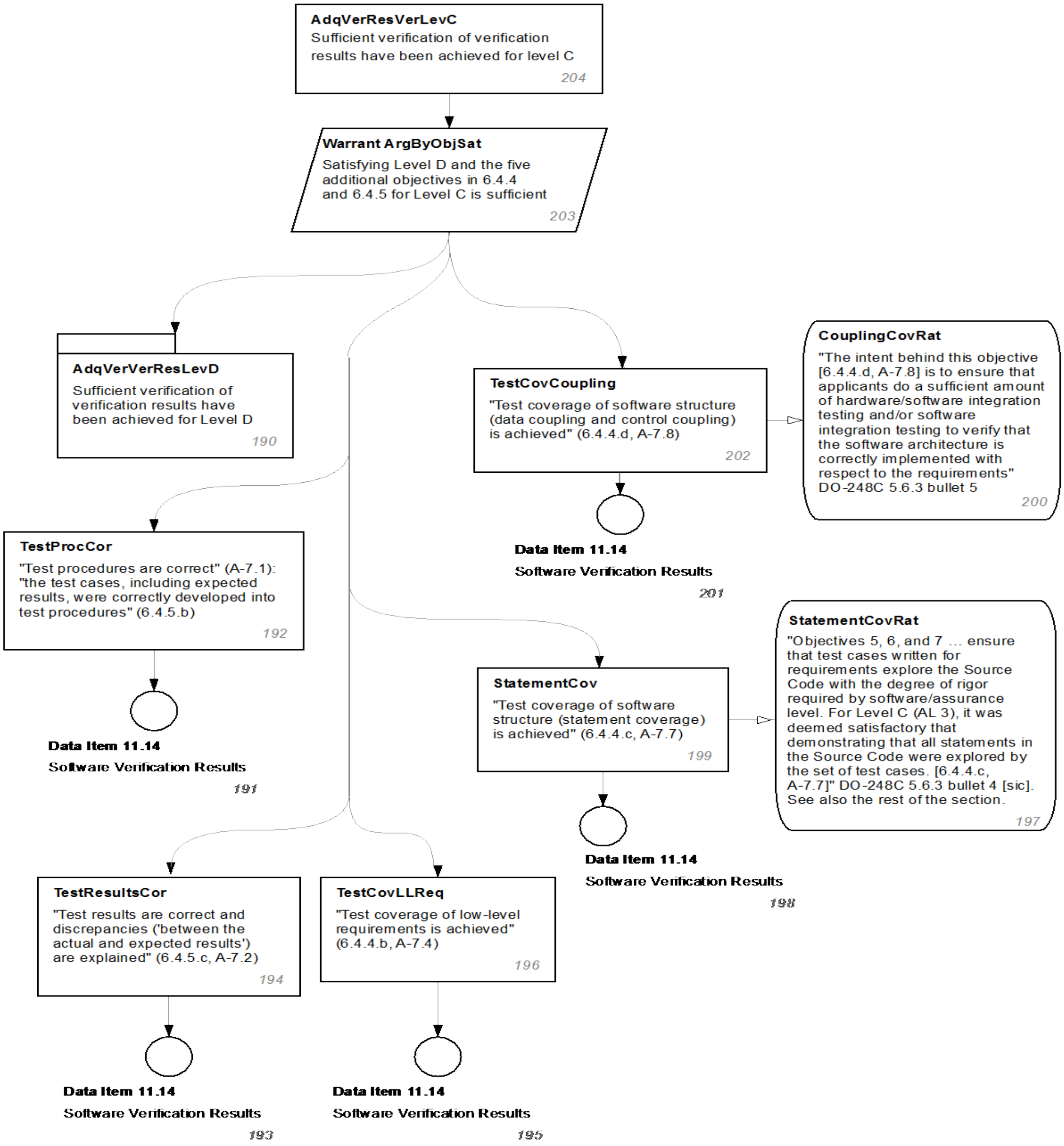

AdqVerResVerLevC
Sufficient verification of verification results have been achieved for level C
204
Warrant ArgByObjSat
Satisfying Level D and the five additional objectives in 6.4.4 and 6.4.5 for Level C is sufficient
203
AdqVerVerResLevD
Sufficient verification of verification results have been achieved for Level D
190
TestCovCoupling
"Test coverage of software structure (data coupling and control coupling) is achieved" (6.4.4.d, A-7.8)
202
CouplingCovRat
"The intent behind this objective [6.4.4.d, A-7.8] is to ensure that applicants do a sufficient amount of hardware/software integration testing and/or software integration testing to verify that the software architecture is correctly implemented with respect to the requirements" DO-248C 5.6.3 bullet 5
200
Data Item 11.14
Software Verification Results
201
TestProcCor
"Test procedures are correct" (A-7.1): "the test cases, including expected results, were correctly developed into test procedures" (6.4.5.b)
192
StatementCovRat
"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 C (AL 3), it was deemed satisfactory that demonstrating that all statements in the Source Code were explored by the set of test cases. [6.4.4.c, A-7.7]" DO-248C 5.6.3 bullet 4 [sic]. See also the rest of the section.
197
StatementCov
"Test coverage of software structure (statement coverage) is achieved" (6.4.4.c, A-7.7)
199
Data Item 11.14
Software Verification Results
191
Data Item 11.14
Software Verification Results
198
TestResultsCor
"Test results are correct and discrepancies ('between the actual and expected results') are explained" (6.4.5.c, A-7.2)
194
TestCovLLReq
"Test coverage of low-level requirements is achieved" (6.4.4.b, A-7.4)
196
Data Item 11.14
Software Verification Results
193
Data Item 11.14
Software Verification Results
195

**Figure 16. Level C: AdqVerVerLevC**

Context element STATEMENTCOVRAT reveals one of the few instances in which a reason is given for certain objectives. Context element STATEMENTCOVRAT reveals another such instance.

Figure 17 expresses the argument concerning the additional refinement steps that are required by Level C and introduces a new element in the notation.

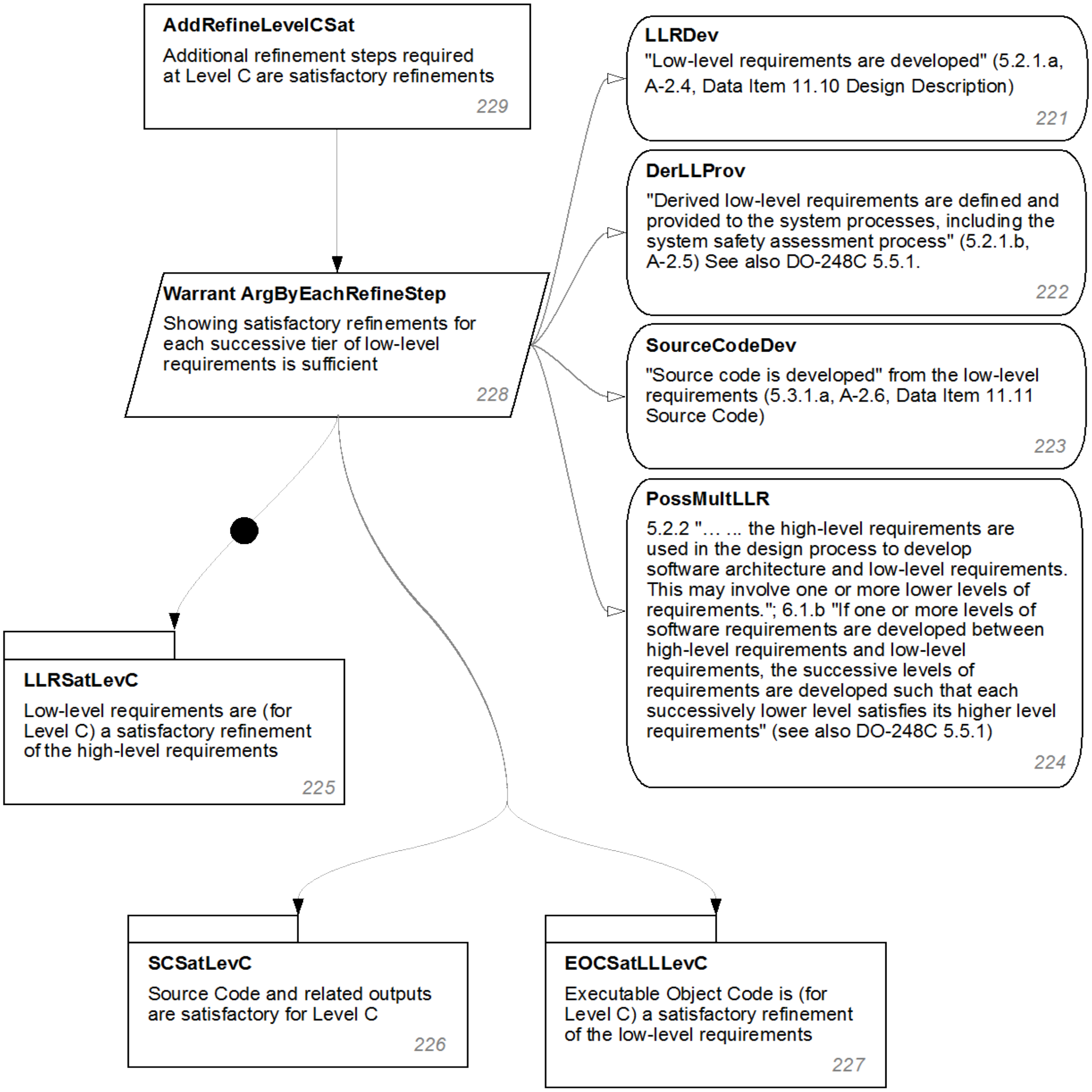

AddRefineLevelCSat
Additional refinement steps required at Level C are satisfactory refinements
229
LLRDev
"Low-level requirements are developed" (5.2.1.a, A-2.4, Data Item 11.10 Design Description)
221
DerLLProv
"Derived low-level requirements are defined and provided to the system processes, including the system safety assessment process" (5.2.1.b, A-2.5) See also DO-248C 5.5.1.
222
Warrant ArgByEachRefineStep
Showing satisfactory refinements for each successive tier of low-level requirements is sufficient
228
SourceCodeDev
"Source code is developed" from the low-level requirements (5.3.1.a, A-2.6, Data Item 11.11 Source Code)
223
PossMultLLR
5.2.2 "... ... the high-level requirements are used in the design process to develop software architecture and low-level requirements. This may involve one or more lower levels of requirements."; 6.1.b "If one or more levels of software requirements are developed between high-level requirements and low-level requirements, the successive levels of requirements are developed such that each successively lower level satisfies its higher level requirements" (see also DO-248C 5.5.1)
224
LLRSatLevC
Low-level requirements are (for Level C) a satisfactory refinement of the high-level requirements
225
SCSatLevC
Source Code and related outputs are satisfactory for Level C
226
EOCSatLLLevC
Executable Object Code is (for Level C) a satisfactory refinement of the low-level requirements
227

**Figure 17. Level C: AddRefineLevelCSat**

The warrant ADDREFINELEVELCSAT takes the general simple form shown previously. Three of the attached context elements concern the existence of certain entities: low-level requirements (LLRDEV), associated derived low-level requirements (DERLLPROV), and source code developed from the low-level requirements (SOURCECODEDEV). The fourth context element (POSSMULTLLR) explains the possibility that more than one tier of low-level requirements may exist.

This possibility of more than one tier is indicated in the diagram by the solid black circle on the directed line connecting the warrant to the module LLRSATLEVC. This module is one of the three premises, along with SCSATLEVC **(Source Code and related outputs are satisfactory for Level C)** and EOCSATLLLEVC **(Executable Object Code is … a satisfactory refinement of the low-**

**level requirements)**. The first and third are elaborated next. The second is available in appendix A of this report.

The argument to establish LLRSatLevC (Low-level requirements are [for Level C] a satisfactory refinement of the high-level requirements) is too large to show in a single figure. Figure 18 shows the top part of the argument, figure 19 shows the part associated with the premise LLRAdqLevelC, and figure 20 shows the part associated with the premise SwArchAdqLevelC.

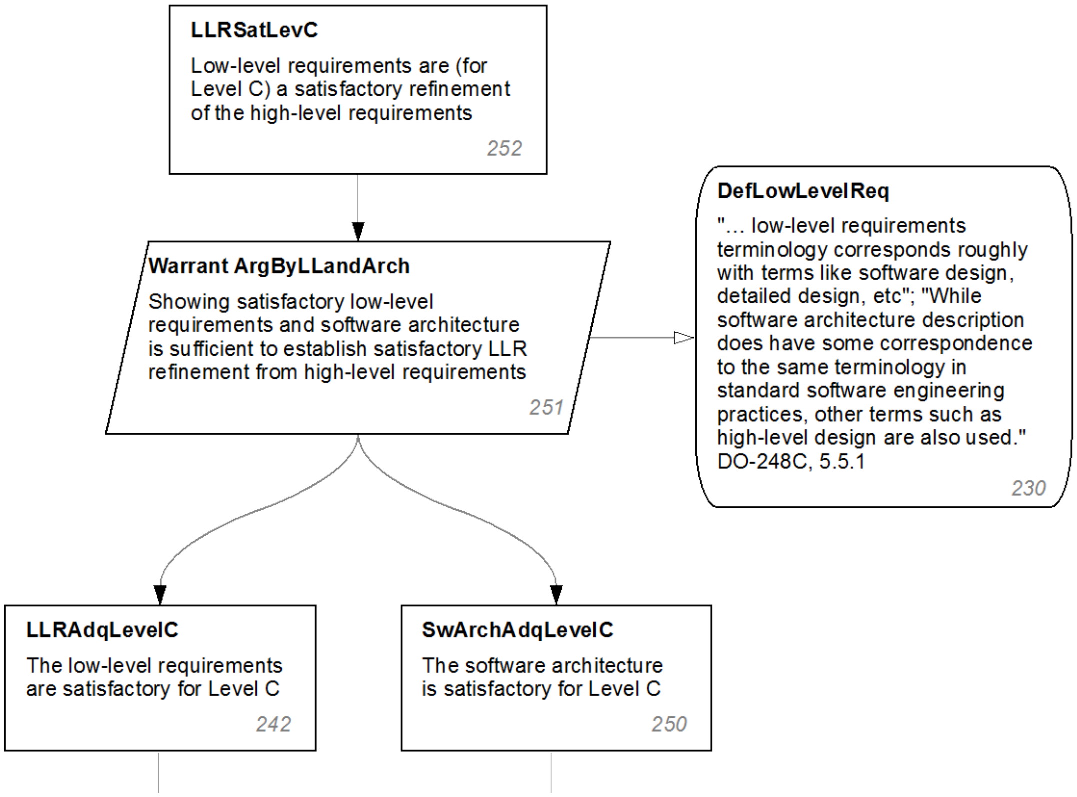

LLRSatLevC
Low-level requirements are (for Level C) a satisfactory refinement of the high-level requirements
252
Warrant ArgByLLandArch
Showing satisfactory low-level requirements and software architecture is sufficient to establish satisfactory LLR refinement from high-level requirements
251
DefLowLevelReq
"... low-level requirements terminology corresponds roughly with terms like software design, detailed design, etc"; "While software architecture description does have some correspondence to the same terminology in standard software engineering practices, other terms such as high-level design are also used." DO-248C, 5.5.1
230
LLRAdqLevelC
The low-level requirements are satisfactory for Level C
242
SwArchAdqLevelC
The software architecture is satisfactory for Level C
250

**Figure 18. Level C: LLRSatLevC (top)**

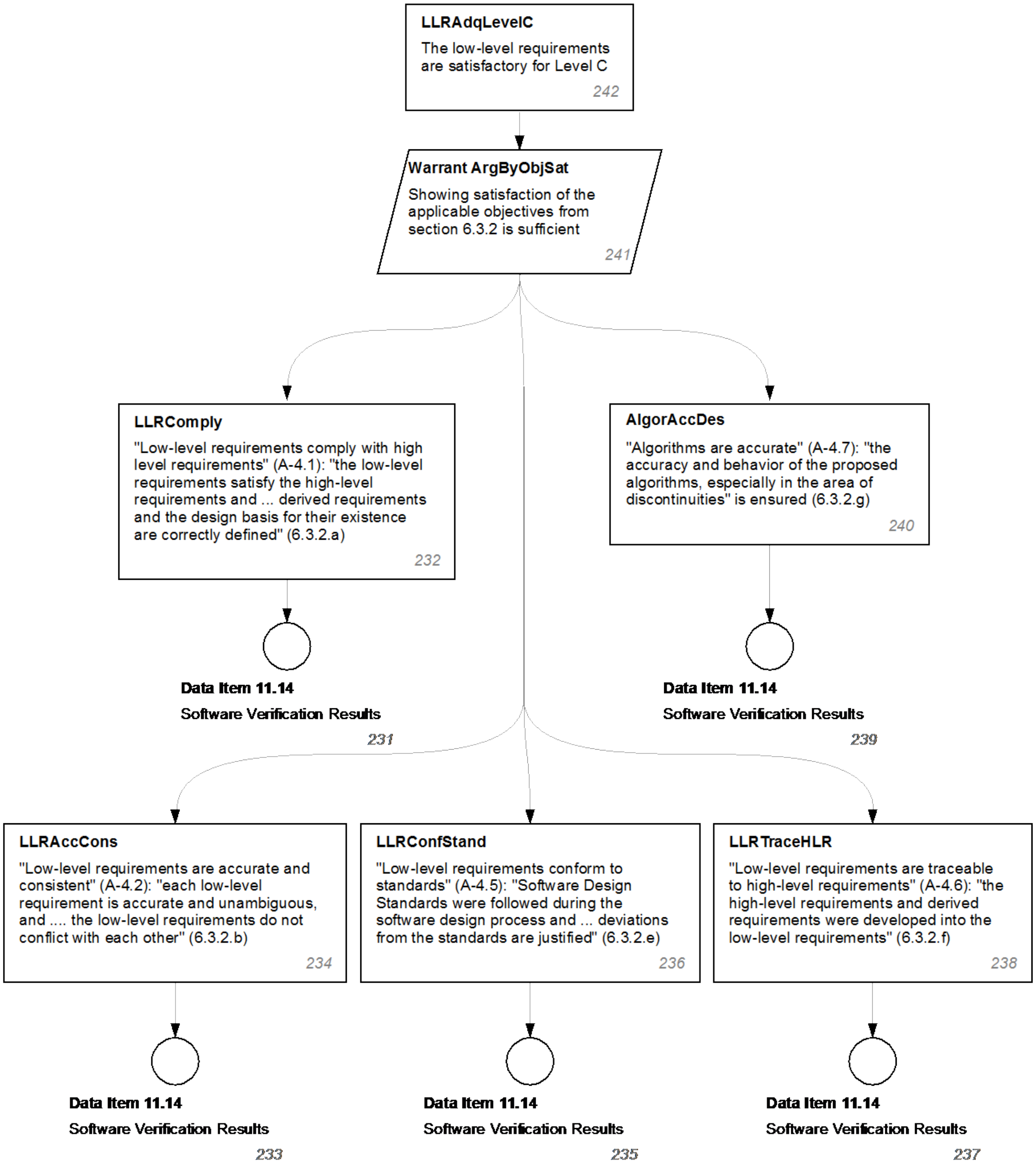

LLRAdqLevelC
The low-level requirements are satisfactory for Level C
242
Warrant ArgByObjSat
Showing satisfaction of the applicable objectives from section 6.3.2 is sufficient
241
LLRComply
"Low-level requirements comply with high level requirements" (A-4.1): "the low-level requirements satisfy the high-level requirements and ... derived requirements and the design basis for their existence are correctly defined" (6.3.2.a)
232
AlgorAccDes
"Algorithms are accurate" (A-4.7): "the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities" is ensured (6.3.2.g)
240
Data Item 11.14
Software Verification Results
231
Data Item 11.14
Software Verification Results
239
LLRAccCons
"Low-level requirements are accurate and consistent" (A-4.2): "each low-level requirement is accurate and unambiguous, and .... the low-level requirements do not conflict with each other" (6.3.2.b)
234
LLRConfStand
"Low-level requirements conform to standards" (A-4.5): "Software Design Standards were followed during the software design process and ... deviations from the standards are justified" (6.3.2.e)
236
LLRTraceHLR
"Low-level requirements are traceable to high-level requirements" (A-4.6): "the high-level requirements and derived requirements were developed into the low-level requirements" (6.3.2.f)
238
Data Item 11.14
Software Verification Results
233
Data Item 11.14
Software Verification Results
235
Data Item 11.14
Software Verification Results
237

**Figure 19. Level C: LLRSatLevC / LLRAdqLevelC (left)**

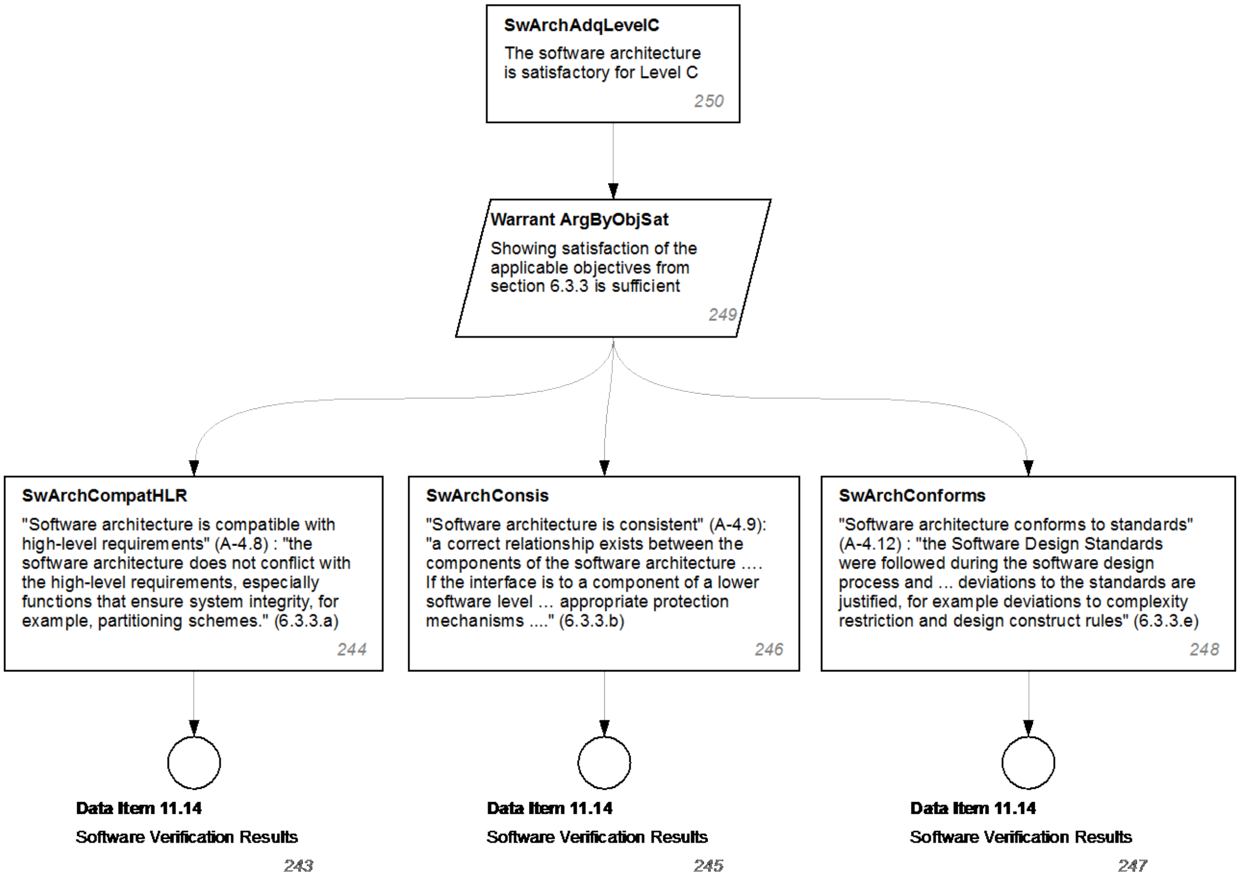

SwArchAdqLevelC
The software architecture is satisfactory for Level C
250
Warrant ArgByObjSat
Showing satisfaction of the applicable objectives from section 6.3.3 is sufficient
249
SwArchCompatHLR
"Software architecture is compatible with high-level requirements" (A-4.8) : "the software architecture does not conflict with the high-level requirements, especially functions that ensure system integrity, for example, partitioning schemes." (6.3.3.a)
244
SwArchConsis
"Software architecture is consistent" (A-4.9): "a correct relationship exists between the components of the software architecture .... If the interface is to a component of a lower software level ... appropriate protection mechanisms ...." (6.3.3.b)
246
SwArchConforms
"Software architecture conforms to standards" (A-4.12) : "the Software Design Standards were followed during the software design process and ... deviations to the standards are justified, for example deviations to complexity restriction and design construct rules" (6.3.3.e)
248
Data Item 11.14
Software Verification Results
243
Data Item 11.14
Software Verification Results
245
Data Item 11.14
Software Verification Results
247

**Figure 20. Level C: LLRSatLevC / SWArchAdqLevelC (right)**

Figure 21 shows the simple argument justifying the sufficiency of the refinement of executable object code from low-level requirements (EOCSatLLLevC). This conclusion is supported by two premises. EOCCompliesLL asserts compliance with low-level requirements, and EOCRobustLL asserts robustness with them. Both premises are supported by reference to the appropriate material from three data items: Data Item 11.13: Software Verification Cases and Procedures; Data Item 11.14: Software Verification Results; and Data Item 11.21: Trace Data.

The shared context item (CC12conflict) notes an apparent minor conflict within the guidance. For Data Item 11.21: Trace Data at Level C, the control category is specified at a less-stringent level in table A-6 than it is for table A-2 (see [1]). Resolving this conflict is simple: Assume that the higher control category applies throughout.

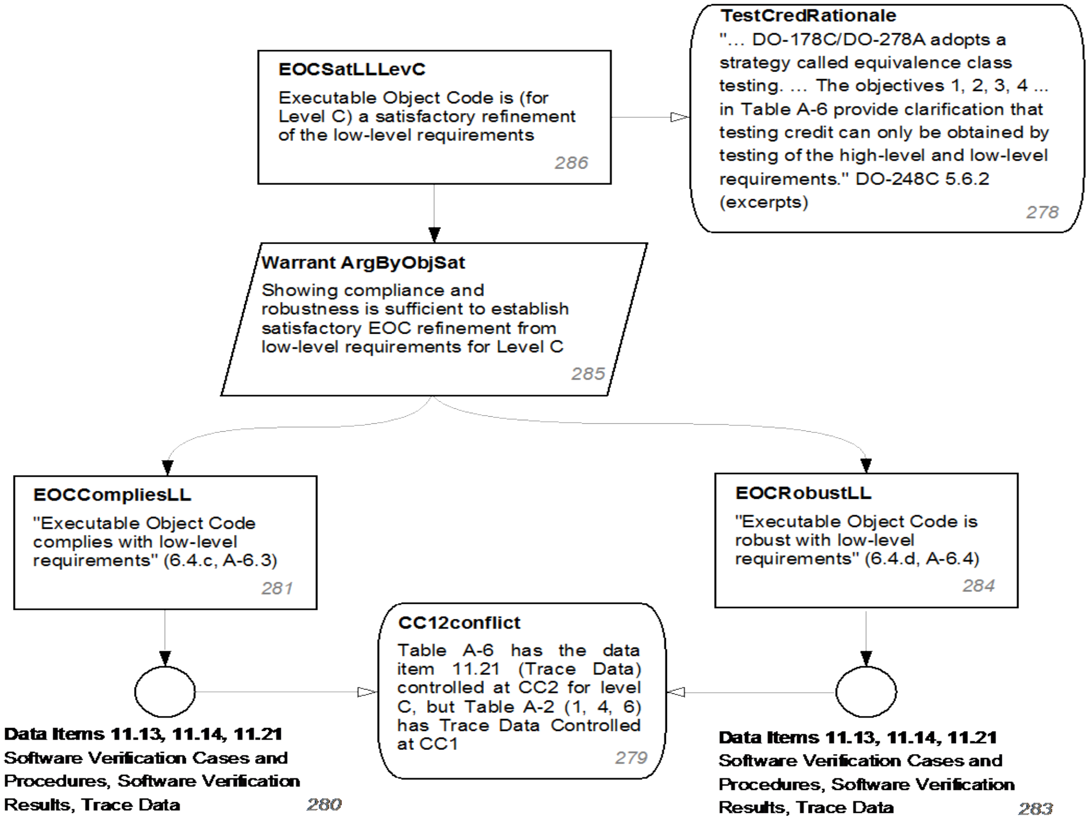

EOCSatLLLevC
Executable Object Code is (for Level C) a satisfactory refinement of the low-level requirements
286
TestCredRationale
"... DO-178C/DO-278A adopts a strategy called equivalence class testing. ... The objectives 1, 2, 3, 4 ... in Table A-6 provide clarification that testing credit can only be obtained by testing of the high-level and low-level requirements." DO-248C 5.6.2 (excerpts)
278
Warrant ArgByObjSat
Showing compliance and robustness is sufficient to establish satisfactory EOC refinement from low-level requirements for Level C
285
EOCCompliesLL
"Executable Object Code complies with low-level requirements" (6.4.c, A-6.3)
281
EOCRobustLL
"Executable Object Code is robust with low-level requirements" (6.4.d, A-6.4)
284
CC12conflict
Table A-6 has the data item 11.21 (Trace Data) controlled at CC2 for level C, but Table A-2 (1, 4, 6) has Trace Data Controlled at CC1
279
Data Items 11.13, 11.14, 11.21
Software Verification Cases and Procedures, Software Verification Results, Trace Data
280
Data Items 11.13, 11.14, 11.21
Software Verification Cases and Procedures, Software Verification Results, Trace Data
283

**Figure 21. Level C: EOCSatLLevC**

## 3.4 LEVEL B ARGUMENTS

For Level B, we will present only four of the arguments from the assurance case:

- SwAcceptableLevB: **Software performs its intended function at acceptable level of safety for Level B.**
- JustifiedConfidenceLevB: **The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for Level B.**
- IndepSatLevB: **Additional independence requirements for Level B are satisfied.**
- AdqVerVerLevB: **Sufficient verification of verification results has been achieved for Level B.**

The main Level B argument is shown in figure 22. Although it would be possible to use an identical structure here as for Levels D and C, a slightly different structure has been used. The form of the conclusion and associated context items remains the same, but the form of the warrant and premises differs substantially. Choosing a different form emphasizes more explicitly how the guidance itself differentiates between the objectives for Levels C and B.

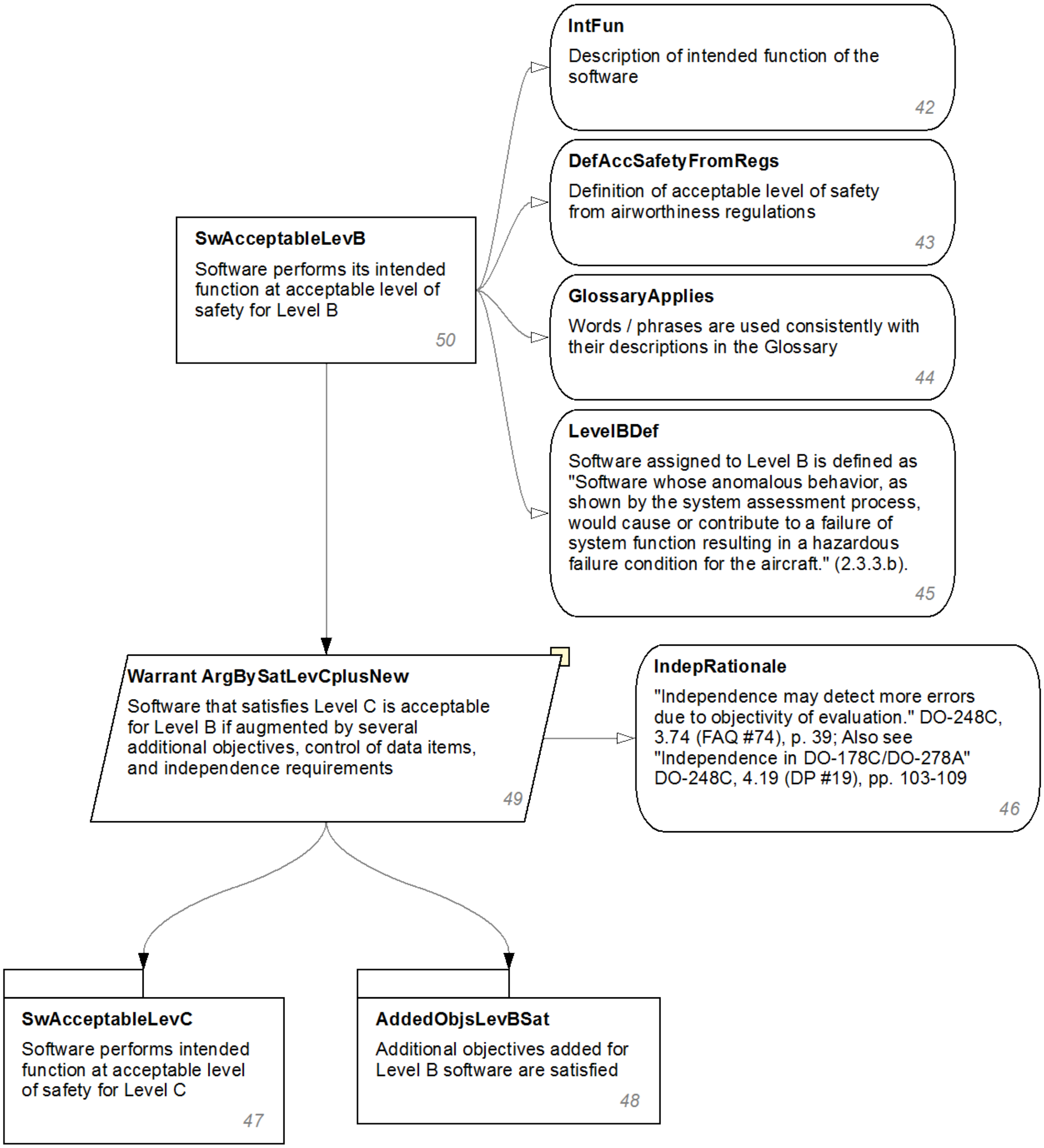

IntFun
Description of intended function of the software
42
DefAccSafetyFromRegs
Definition of acceptable level of safety from airworthiness regulations
43
SwAcceptableLevB
Software performs its intended function at acceptable level of safety for Level B
50
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
44
LevelBDef
Software assigned to Level B is defined as "Software whose anomalous behavior, as shown by the system assessment process, would cause or contribute to a failure of system function resulting in a hazardous failure condition for the aircraft." (2.3.3.b).
45
Warrant ArgBySatLevCplusNew
Software that satisfies Level C is acceptable for Level B if augmented by several additional objectives, control of data items, and independence requirements
49
IndepRationale
"Independence may detect more errors due to objectivity of evaluation." DO-248C, 3.74 (FAQ #74), p. 39; Also see "Independence in DO-178C/DO-278A" DO-248C, 4.19 (DP #19), pp. 103-109
46
SwAcceptableLevC
Software performs intended function at acceptable level of safety for Level C
47
AddedObjsLevBSat
Additional objectives added for Level B software are satisfied
48

**Figure 22. Level B: SwAcceptableLevB**

The warrant (ARGBYSATLEVCPLUSNEW) explicitly identifies the three categories of differences between the guidance for Level C and Level B: several additional objectives, control of data items, and independence requirements. Context element INDEPRATIONALE repeats the assertion from DO-248C concerning the potential efficacy of independence requirements. The details regarding one of the two premises has already been shown (SWACCEPTABLELEVC). The other premise (ADDEDOBJSLEVBSAT) is supported by a simple, six-premise argument enumerating the three added objectives concerning compatibility with the target computer and the three added objectives concerning verifiability. The diagram is shown in appendix A.

The rest of the differences between the guidance for Level C and the guidance for Level B appear in the confidence argument shown in figure 23.

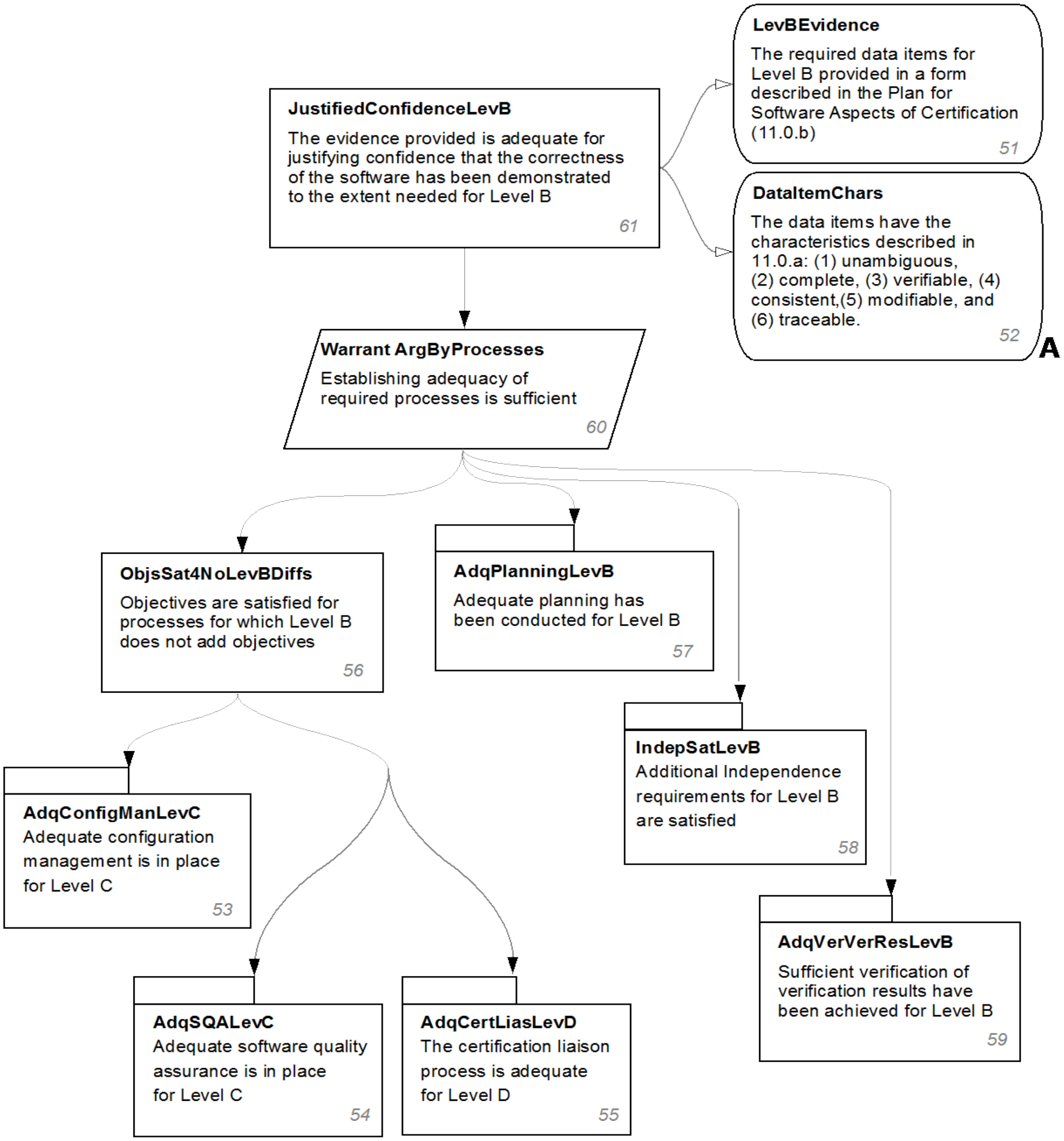

LevBEvidence
The required data items for Level B provided in a form described in the Plan for Software Aspects of Certification (11.0.b)
51
JustifiedConfidenceLevB
The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for Level B
61
DataItemChars
The data items have the characteristics described in 11.0.a: (1) unambiguous, (2) complete, (3) verifiable, (4) consistent,(5) modifiable, and (6) traceable.
52
A
Warrant ArgByProcesses
Establishing adequacy of required processes is sufficient
60
ObjsSat4NoLevBDiffs
Objectives are satisfied for processes for which Level B does not add objectives
56
AdqPlanningLevB
Adequate planning has been conducted for Level B
57
IndepSatLevB
Additional Independence requirements for Level B are satisfied
58
AdqConfigManLevC
Adequate configuration management is in place for Level C
53
AdqVerVerResLevB
Sufficient verification of verification results have been achieved for Level B
59
AdqSQALevC
Adequate software quality assurance is in place for Level C
54
AdqCertLiasLevD
The certification liaison process is adequate for Level D
55

**Figure 23. Level B: JustifiedConfidenceLevB**

The left-hand premise (ObjsSat4NoLevBDiffs) is included in the diagram only for emphasis. The three items beneath it could be directly connected to the warrant without any change in meaning. The three items could, and probably should, be eliminated entirely.[8]

The other three premises encapsulate the additional objectives added for Level B software, which encompass planning, independence requirements, and verification of verification results. The supporting argument establishing Adequate planning has been conducted for Level B is not shown here (see appendix A).

The argument for IndepSatLevB is shown in figure 24. It is not complicated, but it does contain more premises than any argument seen so far:

- HLRComplyInd: **"High-level requirements comply with system requirements" has been shown with independence (6.6.1a, A-3.1).**
- HLRAccConsInd: **"High-level requirements are accurate and consistent" has been shown with independence (6.3.1.b, A-3.2).**
- AlgorAccReqInd: **"Algorithms are accurate" has been shown with independence (6.3.1.g, A-3.7).**
- LLRComplyInd: **"Low-level requirements comply with high level requirements" has been shown with independence (6.3.2.a, A-4.1).**
- LLRAccConsInd: **"Low-level requirements are accurate and consistent" has been shown with independence (6.3.2.b, A-4.2).**
- AlgorAccDesInd: **"Algorithms are accurate" has been shown with independence (6.3.2.g, A-4.7).**
- SCcompLLInd: **"Source Code complies with low-level requirements" has been shown with independence (6.3.4.a, A-5.1).**
- PDICorInd: **"Parameter Data Item File is correct and complete" has been shown with independence (6.6.a, A-5.8).**
- PDIVerInd: **"Verification of Parameter Data Item File is achieved" with independence (6.6.b, A-5.9).**
- EOCCompliesLLInd: **"Executable Object Code complies with low-level requirements" has been shown with independence (6.4.c, A-6.3).**

The context elements attached to the warrant are explanatory only. They could be removed without any change in the meaning of the argument.

[8] AdqConfigManLevC, AdqSQALevC, and AdqCertLiasLevD do not need to be repeated here because they are included as part of the confidence argument for Level C (which is included in SwAcceptableLevC and is one of the two premises for the main Level B argument). AddRefinementLevelCSat, which is not reproduced in the Level B confidence argument, is also included there.

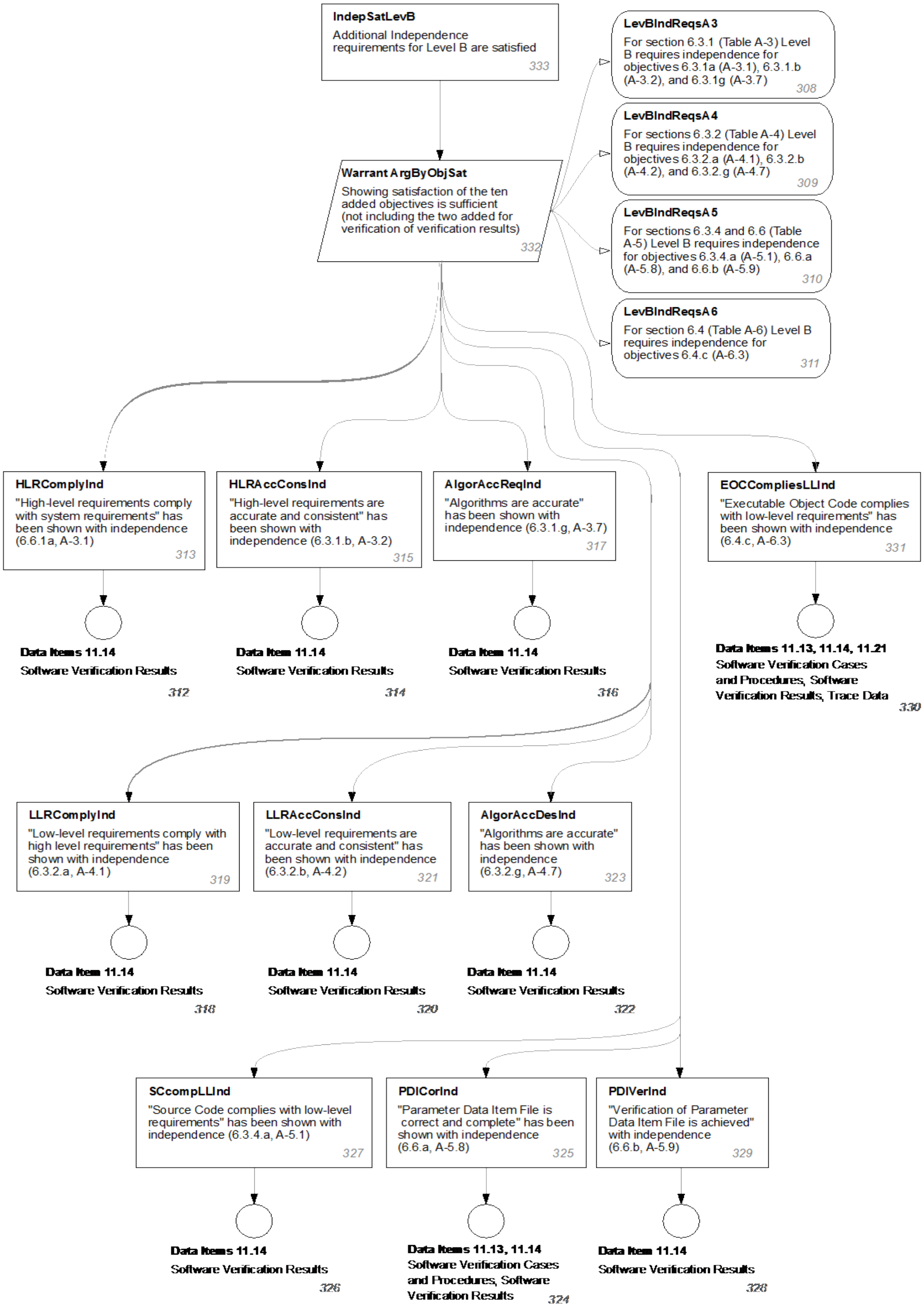

IndepSatLevB
Additional Independence requirements for Level B are satisfied
333
LevBIndReqsA3
For section 6.3.1 (Table A-3) Level B requires independence for objectives 6.3.1a (A-3.1), 6.3.1.b (A-3.2), and 6.3.1g (A-3.7)
308
LevBIndReqsA4
For sections 6.3.2 (Table A-4) Level B requires independence for objectives 6.3.2.a (A-4.1), 6.3.2.b (A-4.2), and 6.3.2.g (A-4.7)
309
Warrant ArgByObjSat
Showing satisfaction of the ten added objectives is sufficient (not including the two added for verification of verification results)
332
LevBIndReqsA5
For sections 6.3.4 and 6.6 (Table A-5) Level B requires independence for objectives 6.3.4.a (A-5.1), 6.6.a (A-5.8), and 6.6.b (A-5.9)
310
LevBIndReqsA6
For section 6.4 (Table A-6) Level B requires independence for objectives 6.4.c (A-6.3)
311
HLRComplyInd
"High-level requirements comply with system requirements" has been shown with independence (6.6.1a, A-3.1)
313
HLRAccConsInd
"High-level requirements are accurate and consistent" has been shown with independence (6.3.1.b, A-3.2)
315
AlgorAccReqInd
"Algorithms are accurate" has been shown with independence (6.3.1.g, A-3.7)
317
EOCCompliesLLInd
"Executable Object Code complies with low-level requirements" has been shown with independence (6.4.c, A-6.3)
331
Data Items 11.14
Software Verification Results
312
Data Item 11.14
Software Verification Results
314
Data Item 11.14
Software Verification Results
316
Data Items 11.13, 11.14, 11.21
Software Verification Cases and Procedures, Software Verification Results, Trace Data
330
LLRComplyInd
"Low-level requirements comply with high level requirements" has been shown with independence (6.3.2.a, A-4.1)
319
LLRAccConsInd
"Low-level requirements are accurate and consistent" has been shown with independence (6.3.2.b, A-4.2)
321
AlgorAccDesInd
"Algorithms are accurate" has been shown with independence (6.3.2.g, A-4.7)
323
Data Item 11.14
Software Verification Results
318
Data Item 11.14
Software Verification Results
320
Data Item 11.14
Software Verification Results
322
SCcompLLInd
"Source Code complies with low-level requirements" has been shown with independence (6.3.4.a, A-5.1)
327
PDICorInd
"Parameter Data Item File is correct and complete" has been shown with independence (6.6.a, A-5.8)
325
PDIVerInd
"Verification of Parameter Data Item File is achieved" with independence (6.6.b, A-5.9)
329
Data Items 11.14
Software Verification Results
326
Data Items 11.13, 11.14
Software Verification Cases and Procedures, Software Verification Results
324
Data Item 11.14
Software Verification Results
328

**Figure 24. Level B: IndepSatLevB**

Two independence requirements added for Level B were not included. Because these two requirements are associated with the verification of verification results, they are enumerated in the argument for ADQVERVERLEVB , which is shown in figure 25.

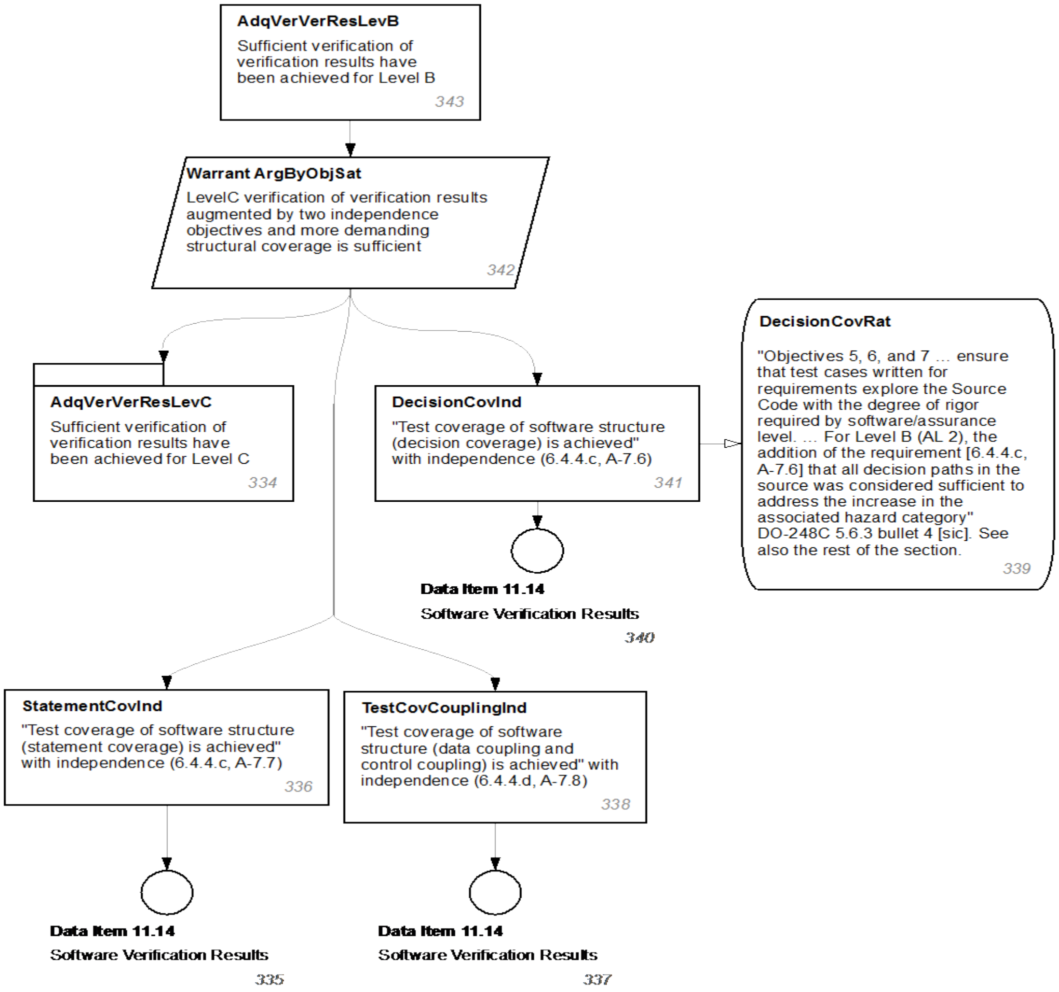

AdqVerVerResLevB
Sufficient verification of verification results have been achieved for Level B
343
Warrant ArgByObjSat
LevelC verification of verification results augmented by two independence objectives and more demanding structural coverage is sufficient
342
AdqVerVerResLevC
Sufficient verification of verification results have been achieved for Level C
334
DecisionCovInd
"Test coverage of software structure (decision coverage) is achieved" with independence (6.4.4.c, A-7.6)
341
DecisionCovRat
"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 B (AL 2), the addition of the requirement [6.4.4.c, A-7.6] that all decision paths in the source was considered sufficient to address the increase in the associated hazard category" DO-248C 5.6.3 bullet 4 [sic]. See also the rest of the section.
339
Data Item 11.14
Software Verification Results
340
StatementCovInd
"Test coverage of software structure (statement coverage) is achieved" with independence (6.4.4.c, A-7.7)
336
TestCovCouplingInd
"Test coverage of software structure (data coupling and control coupling) is achieved" with independence (6.4.4.d, A-7.8)
338
Data Item 11.14
Software Verification Results
335
Data Item 11.14
Software Verification Results
337

**Figure 25. Level B: AdqVerVerLevB**

For Level B verification of verification results, DO-178C requirements independent achievement of statement (STATEMENTCOVIND) and coupling coverage (TESTCOVCOUPLINGIND). It also adds a requirement for decision coverage, which must be achieved with independence (DECISIONCOVIND). The evidence is contained in the relevant parts of DATA ITEM 11.14: Software Verification Results.

## 3.5 LEVEL A ARGUMENTS

Explicating the Level A guidance required only four arguments: the main argument, the confidence argument, and two supporting arguments for premises in the confidence argument. The main argument is shown in figure 26. Besides including the context elements shown for every lower level and an associated confidence argument, it consists solely of incorporating the Level B conclusion and all arguments supporting that conclusion. As explained in LEVALEVBDIFFSCONF, this is because **all the differences between objectives for Level A and Level B address matters of confidence.**

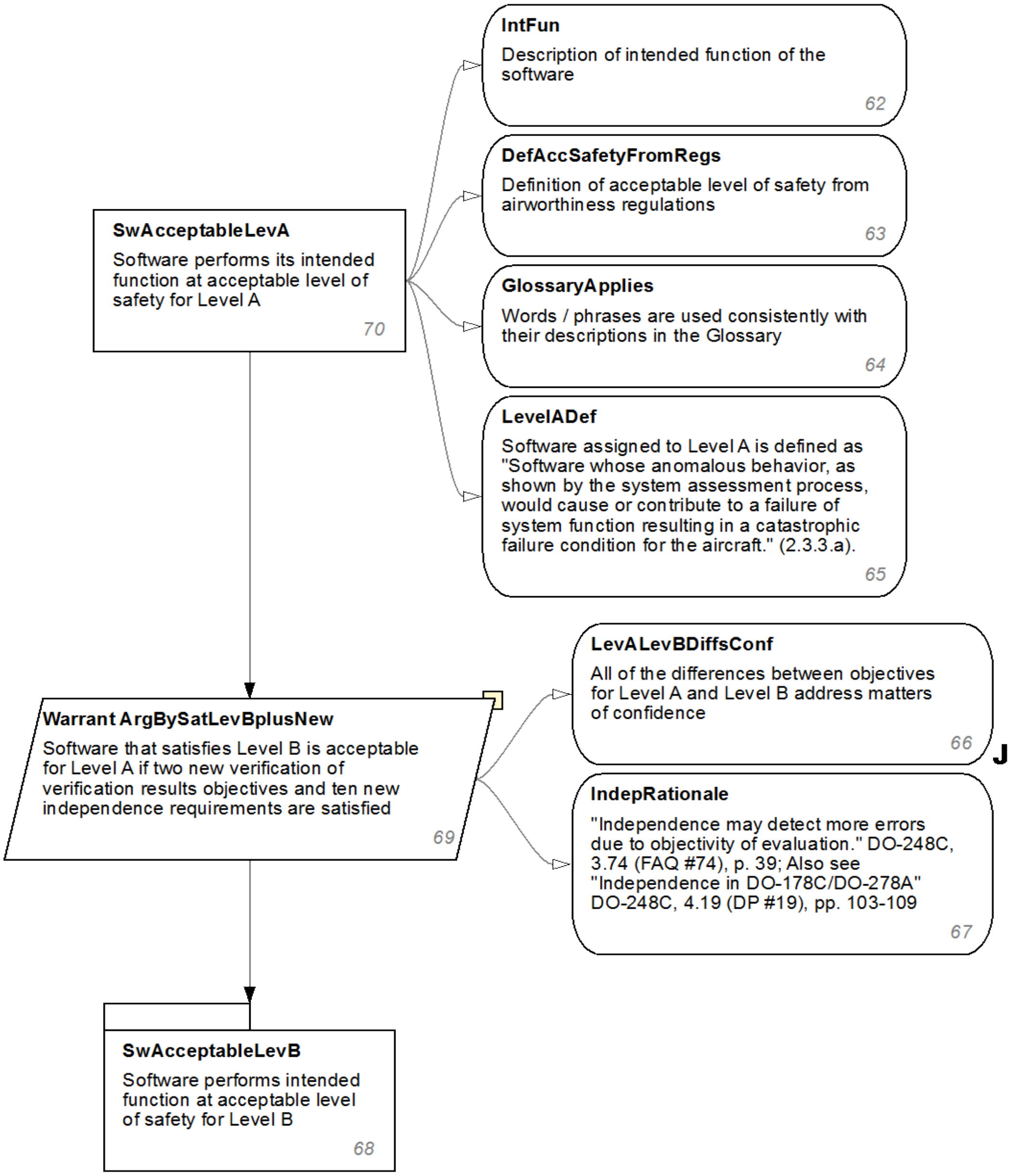

IntFun
Description of intended function of the software
62
DefAccSafetyFromRegs
Definition of acceptable level of safety from airworthiness regulations
63
SwAcceptableLevA
Software performs its intended function at acceptable level of safety for Level A
70
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
64
LevelADef
Software assigned to Level A is defined as "Software whose anomalous behavior, as shown by the system assessment process, would cause or contribute to a failure of system function resulting in a catastrophic failure condition for the aircraft." (2.3.3.a).
65
LevALevBDiffsConf
All of the differences between objectives for Level A and Level B address matters of confidence
66
J
Warrant ArgBySatLevBplusNew
Software that satisfies Level B is acceptable for Level A if two new verification of verification results objectives and ten new independence requirements are satisfied
69
IndepRationale
"Independence may detect more errors due to objectivity of evaluation." DO-248C, 3.74 (FAQ #74), p. 39; Also see "Independence in DO-178C/DO-278A" DO-248C, 4.19 (DP #19), pp. 103-109
67
SwAcceptableLevB
Software performs intended function at acceptable level of safety for Level B
68

**Figure 26. Level A: SWAcceptableLevA**

These matters of confidence are explicated in the argument for JUSTIFIEDCONFIDENCELEVA, which is shown in figure 27. The explanatory premise OBJSSAT4NOLEVADIFFS, like its analogous premise in the Level B confidence argument, is not strictly necessary. The premises INDEPSATLEVA and ADQVERVERLEVA are necessary. The former is shown in figure 28; the latter, because of its size, is shown in figures 29–31. By this point, these arguments should be understood without any additional explanatory text. Their form and content are directly analogous to the similar arguments for Level B.

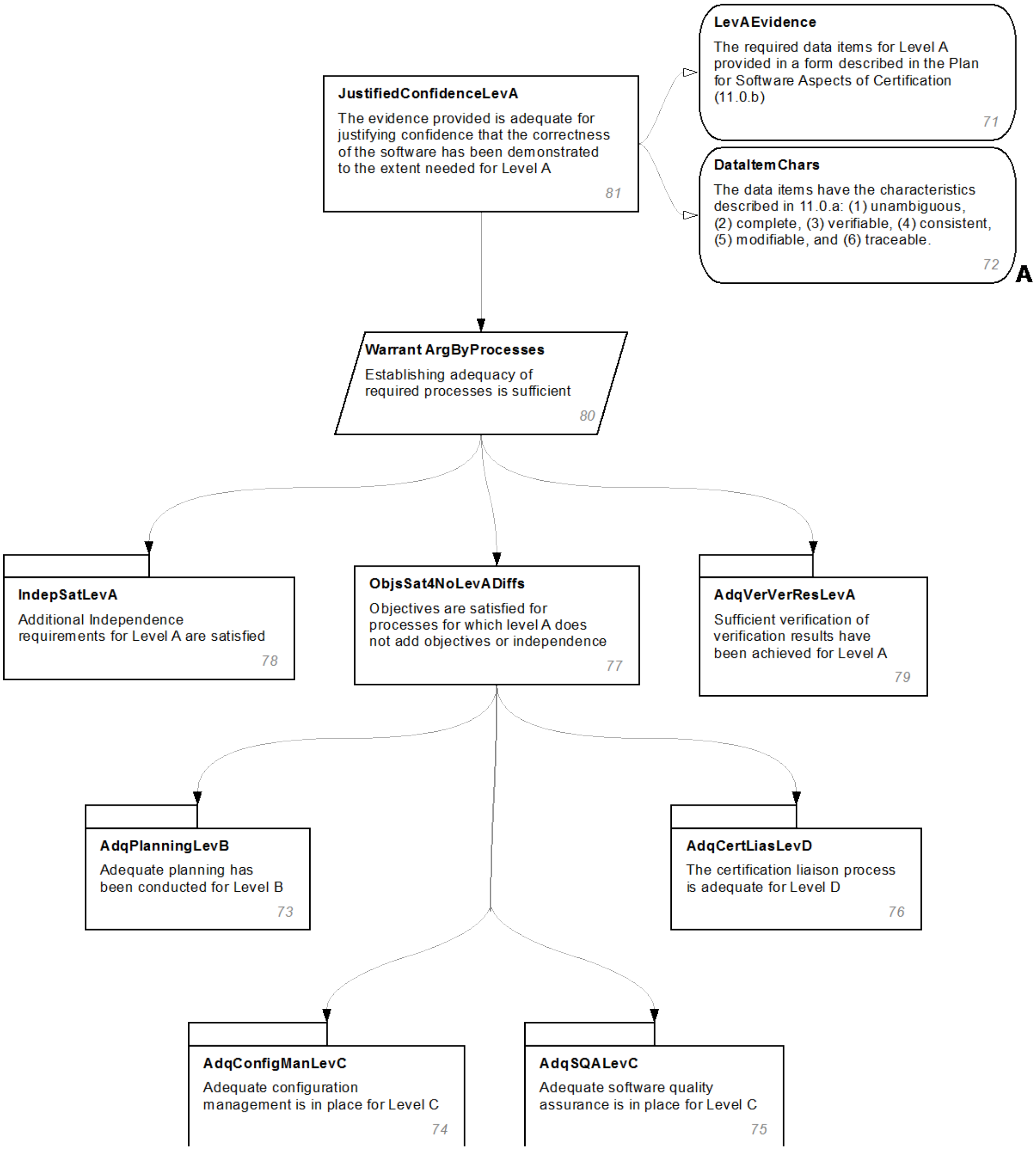

LevAEvidence
The required data items for Level A provided in a form described in the Plan for Software Aspects of Certification (11.0.b)
71
DataItemChars
The data items have the characteristics described in 11.0.a: (1) unambiguous, (2) complete, (3) verifiable, (4) consistent, (5) modifiable, and (6) traceable.
72
A
JustifiedConfidenceLevA
The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for Level A
81
Warrant ArgByProcesses
Establishing adequacy of required processes is sufficient
80
IndepSatLevA
Additional Independence requirements for Level A are satisfied
78
ObjsSat4NoLevADiffs
Objectives are satisfied for processes for which level A does not add objectives or independence
77
AdqVerVerResLevA
Sufficient verification of verification results have been achieved for Level A
79
AdqPlanningLevB
Adequate planning has been conducted for Level B
73
AdqCertLiasLevD
The certification liaison process is adequate for Level D
76
AdqConfigManLevC
Adequate configuration management is in place for Level C
74
AdqSQALevC
Adequate software quality assurance is in place for Level C
75

**Figure 27. Level A: JustifiedConfidenceLevA**

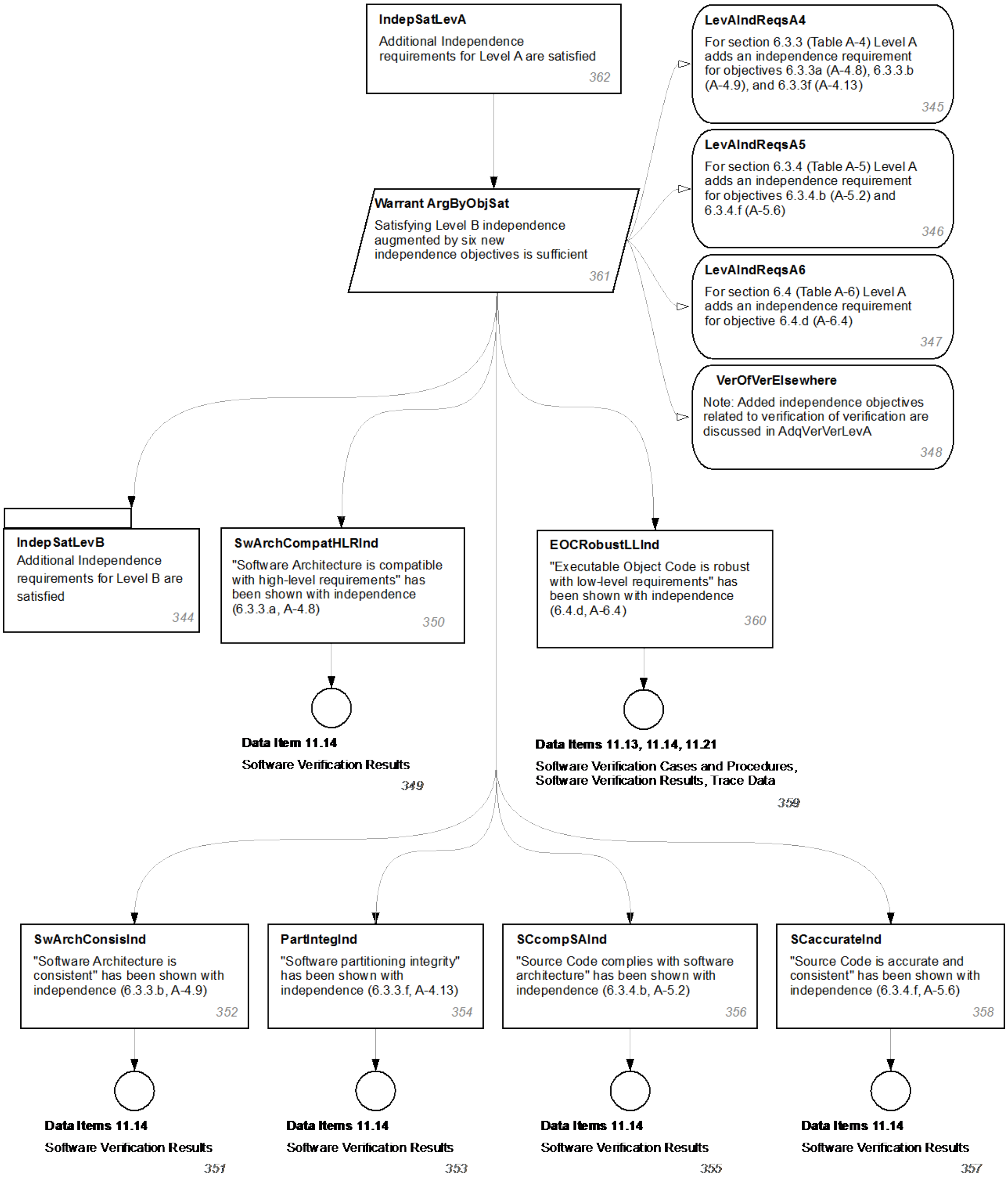

IndepSatLevA
Additional Independence requirements for Level A are satisfied
362
LevAIndReqsA4
For section 6.3.3 (Table A-4) Level A adds an independence requirement for objectives 6.3.3a (A-4.8), 6.3.3.b (A-4.9), and 6.3.3f (A-4.13)
345
LevAIndReqsA5
For section 6.3.4 (Table A-5) Level A adds an independence requirement for objectives 6.3.4.b (A-5.2) and 6.3.4.f (A-5.6)
346
LevAIndReqsA6
For section 6.4 (Table A-6) Level A adds an independence requirement for objective 6.4.d (A-6.4)
347
VerOfVerElsewhere
Note: Added independence objectives related to verification of verification are discussed in AdqVerVerLevA
348
Warrant ArgByObjSat
Satisfying Level B independence augmented by six new independence objectives is sufficient
361
IndepSatLevB
Additional Independence requirements for Level B are satisfied
344
SwArchCompatHLRInd
"Software Architecture is compatible with high-level requirements" has been shown with independence (6.3.3.a, A-4.8)
350
EOCRobustLLInd
"Executable Object Code is robust with low-level requirements" has been shown with independence (6.4.d, A-6.4)
360
Data Item 11.14
Software Verification Results
349
Data Items 11.13, 11.14, 11.21
Software Verification Cases and Procedures, Software Verification Results, Trace Data
359
SwArchConsisInd
"Software Architecture is consistent" has been shown with independence (6.3.3.b, A-4.9)
352
PartIntegInd
"Software partitioning integrity" has been shown with independence (6.3.3.f, A-4.13)
354
SCcompSAInd
"Source Code complies with software architecture" has been shown with independence (6.3.4.b, A-5.2)
356
SCaccurateInd
"Source Code is accurate and consistent" has been shown with independence (6.3.4.f, A-5.6)
358
Data Items 11.14
Software Verification Results
351
Data Items 11.14
Software Verification Results
353
Data Items 11.14
Software Verification Results
355
Data Items 11.14
Software Verification Results
357

**Figure 28. Level A: IndepSatLevA**

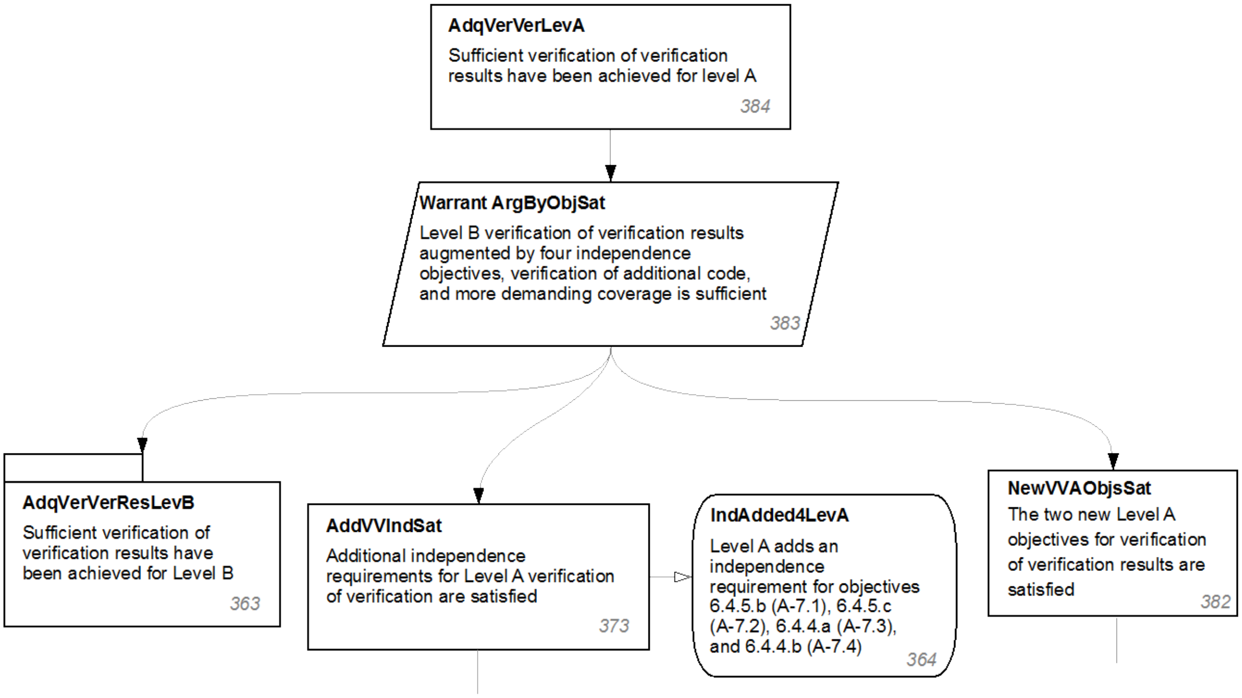

AdqVerVerLevA
Sufficient verification of verification results have been achieved for level A
384
Warrant ArgByObjSat
Level B verification of verification results augmented by four independence objectives, verification of additional code, and more demanding coverage is sufficient
383
AdqVerVerResLevB
Sufficient verification of verification results have been achieved for Level B
363
AddVVIndSat
Additional independence requirements for Level A verification of verification are satisfied
373
IndAdded4LevA
Level A adds an independence requirement for objectives 6.4.5.b (A-7.1), 6.4.5.c (A-7.2), 6.4.4.a (A-7.3), and 6.4.4.b (A-7.4)
364
NewVVAObjsSat
The two new Level A objectives for verification of verification results are satisfied
382

**Figure 29. Level A: AdqVerVerLevA (top)**

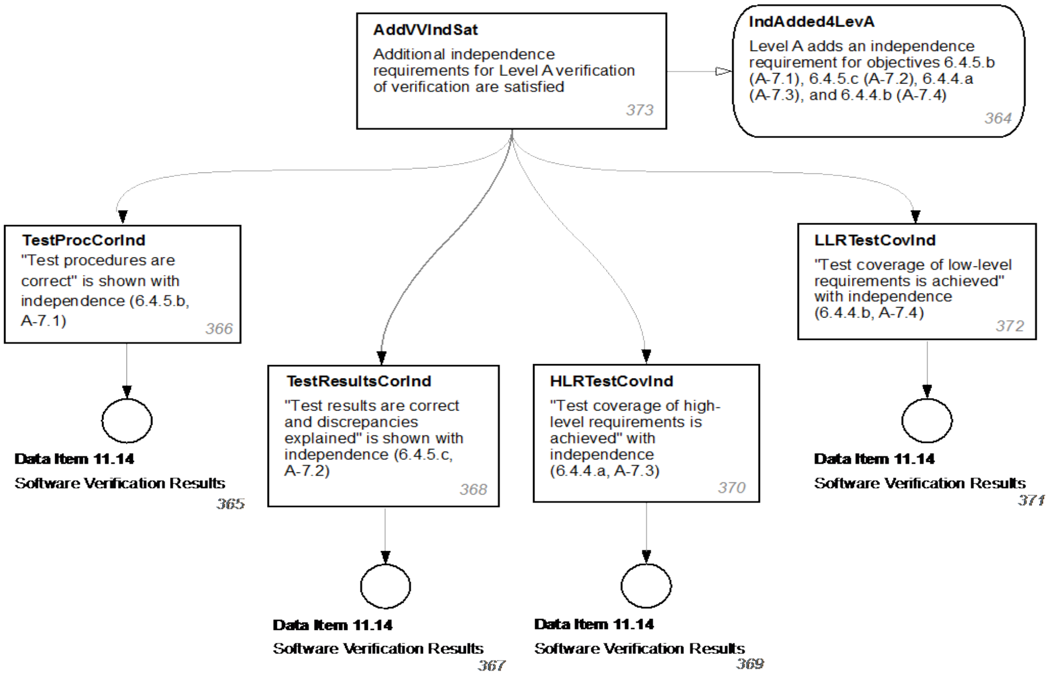

AddVVIndSat
Additional independence requirements for Level A verification of verification are satisfied
373
IndAdded4LevA
Level A adds an independence requirement for objectives 6.4.5.b (A-7.1), 6.4.5.c (A-7.2), 6.4.4.a (A-7.3), and 6.4.4.b (A-7.4)
364
TestProcCorInd
"Test procedures are correct" is shown with independence (6.4.5.b, A-7.1)
366
TestResultsCorInd
"Test results are correct and discrepancies explained" is shown with independence (6.4.5.c, A-7.2)
368
HLRTestCovInd
"Test coverage of high-level requirements is achieved" with independence (6.4.4.a, A-7.3)
370
LLRTestCovInd
"Test coverage of low-level requirements is achieved" with independence (6.4.4.b, A-7.4)
372
Data Item 11.14
Software Verification Results
365
Data Item 11.14
Software Verification Results
367
Data Item 11.14
Software Verification Results
369
Data Item 11.14
Software Verification Results
371

**Figure 30. Level A: AdqVerVerLevA (left)**

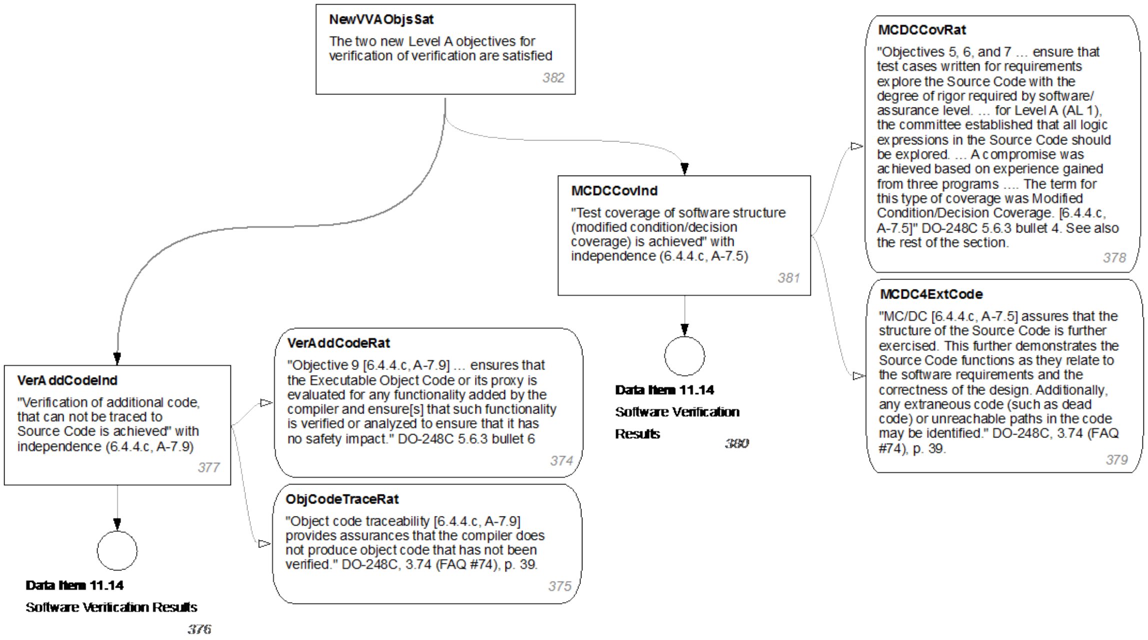

NewVVAObjsSat
The two new Level A objectives for verification of verification are satisfied
382
MCDCCovRat
"Objectives 5, 6, and 7 ... ensure that test cases written for requirements explore the Source Code with the degree of rigor required by software/ assurance level. ... for Level A (AL 1), the committee 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. [6.4.4.c, A-7.5]" DO-248C 5.6.3 bullet 4. See also the rest of the section.
378
MCDCCovInd
"Test coverage of software structure (modified condition/decision coverage) is achieved" with independence (6.4.4.c, A-7.5)
381
MCDC4ExtCode
"MC/DC [6.4.4.c, A-7.5] 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." DO-248C, 3.74 (FAQ #74), p. 39.
379
Data Item 11.14
Software Verification
Results
380
VerAddCodeRat
"Objective 9 [6.4.4.c, A-7.9] ... 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
374
VerAddCodeInd
"Verification of additional code, that can not be traced to Source Code is achieved" with independence (6.4.4.c, A-7.9)
377
ObjCodeTraceRat
"Object code traceability [6.4.4.c, A-7.9] provides assurances that the compiler does not produce object code that has not been verified." DO-248C, 3.74 (FAQ #74), p. 39.
375
Data Item 11.14
Software Verification Results
376

**Figure 31. Level A: AdqVerVerLevA (right)**

## 4. OBSERVATIONS AND ANALYSIS

Two distinct aspects of the created assurance case deserve evaluation: the fidelity of the case to the guidance it purports to explicate and the adequacy of the case for providing the desired assurance. In evaluating the fidelity of the case, an answer is attempted regarding whether the case properly captures the guidance contained in DO-178-C. In evaluating the adequacy of the case, an answer is attempted regarding whether DO-178C meets its intended purpose. Unless the answer to the first question is "yes," the developed assurance case cannot be used legitimately to attempt to answer the second question. Therefore, fidelity is considered first.

### 4.1 ABOUT FIDELITY

During the course of the Explicate '78 research, independent evaluations of the fidelity of the case were sought on multiple occasions. Early in the work, the FAA's former Chief Scientist and Technical Advisor for Software provided a comprehensive critique of the first version of the Level D arguments and offered suggestions for modifications to the first version of the Level C arguments (including concurring with the then nascent notion of placing the arguments concerning low-level requirements in the confidence argument). He and other FAA personnel also provided helpful comments when the initial versions of the Level B and Level A arguments were created and as subsequent modifications were made to the arguments for all levels. Non-FAA personnel were also invited to comment when the work was presented in public forums, particularly the 2014 National Systems, Software, and Airborne Electronic Hardware Conference, and the 2015 Safety-Critical Systems Symposium.

In the public forums, no one expressed any strong doubts about the overall fidelity of the arguments, but several people asked why certain choices were made. The three most common questions concerned matters that have already been explained: Why activities are not included, why certain objectives are expressed as context instead of as conclusions to be demonstrated, and why low-level requirements are contained in the confidence argument instead of in the primary argument.

Two other questions were also raised: one concerning why entire data items are cited as evidence instead of just the relevant parts of the items and the other about why section 12 matters are mentioned only once. The same response answers both questions: These choices enable the argument to more accurately represent the DO-178C guidance as written. DO-178C itself does not specify specific parts of a data item in reference to objectives but rather the entire item. For section 12, only one objective exists. Including more detail in the assurance case would have violated the faithfulness to the text principle described in section 3.1.

In addition to these multiple requests for external reviews, two comprehensive internal reviews were conducted by two individuals. These reviews uncovered several minor inconsistences and omissions[9], which were subsequently corrected. The version of the arguments presented here incorporates all of these changes.

As a result of these reviews (both external and internal), a plausible inference may be drawn that the assurance case described in this report accurately captures the implicit assurance case underlying DO-178C. Therefore, evaluating the adequacy of the assurance case for purpose can provide insight into the adequacy of DO-178C itself.

## 4.2 ABOUT ADEQUACY

Methods for evaluating the adequacy of assurance cases are a subject of continuing research and debate. References [30, 31, and 32] provide the current thinking on this subject. Educational materials produced separately as part of this project are also available for review. Module 3 specifically addresses evaluation methods.

Two of the most vigorous ongoing debates concern matters that are not pertinent to the Explicate '78 assurance case. One such debate centers on the extent to which it is possible (and desirable) to create assurance cases consisting entirely (or at least, primarily) of deductive arguments.[10] The other vigorous debate involves whether the strength of arguments can (and should) be quantified. Neither of these debates pertain here. Creating a mostly deductive case would have required violating the guiding principles of faithfulness to the text and minimum speculation. Trying to quantify argument strength would have required violating the minimum speculation principle by, for example, necessitating speculation about the appropriate numbers to attach to the efficacy of the means used to satisfy the various objectives. Therefore, neither deductive nor quantitative evaluation is relevant to the DO-178C assurance case. Qualitative evaluation is the only option.

[9] This includes slightly less minor, but not fatal, inconsistency in the previously mentioned Level B arguments.

[10] A *deductive* argument is one in which true premises and a valid structure guarantee the truth of the conclusion.

Several approaches for qualitative evaluation exist, but all of them are essentially variants of Kelly's original four-step process [33–34]. The details of this process are described in educational module 3 and are not repeated here. The four steps may be summarized as: 1) identifying the key elements of the case; 2) checking for structural errors; 3) checking for an appropriate amount of detail; and 4) assessing argument strength. The first three steps were during development and revision of the DO-178C case because these steps directly affect the fidelity of explication. Only after the case was completed in the form shown was step four applied.

Six overall observations arose from the assessment.

### 4.2.1 Observation 1 – Foundational Reliance is Placed on a Separate Safety Process

The main arguments for Levels C and D directly (and for Levels A and B indirectly) rely on a warrant (ArgByCorrectness) that involves supporting a conclusion partially about safety by premises exclusively about correctness. In general, equating safety and correctness is not justified. It *is* justified in the DO-178C context based on the assumption explained in ReqAllocValidSuff; the provision that derived requirements must be provided to a separate system safety assessment process.

Given a set of requirements that eventually includes everything necessary to provide an adequate level of safety, then ensuring that the requirements are met, necessarily ensures that an adequate level of safety is provided. Therefore, the guidance needs to ensure only that the 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 general (safety and correctness are different concepts) but does in the specific environment in which software is built for airborne systems and equipment. The assurance case makes the necessity of the implication holding clear, whereas the textual guidance tends to hide it from view well that some critics of DO-178C (and its predecessors) seem totally unaware of it.

Another way in which the guidance places foundational reliance on a separate safety process is in the assignment of criticality levels to the software. The determination of the software level is not part of the DO-178C guidance; it is part of the separate system safety assessment process. If the software level is assigned improperly, the application of DO-178C is not likely to have the desired results. For example, if software is assigned Level D but its anomalous behavior "would cause or contribute to a failure of system function resulting in a catastrophic failure condition for the aircraft," then the likelihood that critical errors will be introduced and missed is higher than it would have been had the software been properly assigned to Level A. Conversely, if the software is improperly assigned to Level A, when the worst possible outcome of its anomalous behavior would be a "minor failure condition for the aircraft," then resources are likely to have been expended that did not need to be expended.

Looking into the future, the question arises whether reliance on separate processes for safety assessment and software development will continue to be possible. As software becomes more pervasive, and functions are allocated to software, the interconnections between the software and the system may become so great that a more intimate relationship becomes necessary between the two processes.

### 4.2.2 Observation 2 – Foundational Reliance is Placed on System Requirements

Within the airborne systems software community, the foundational reliance on the quality of the system requirements allocated to software is well understood. These requirements are developed outside of DO-178C. As made explicitly clear in the assurance case, DO-178C's guidance is intended to ensure the implementation of these requirements is correct. If the allocated requirements are bad (e.g., they fail to account for certain known potential hazardous states, or, worse, they require the entry into such a state), then following the guidance may prove well in the correct implementation of these bad requirements.

In some other software communities, the reliance may be less understood. This is particularly true for communities in which standard practice involves software developers creating their own requirements. Such a community would not be well-served by adopting DO-178C alone as guidance.

### 4.2.3 Observation 3 – Critical Reliance is Placed on Data Item Integrity

Data items are cited as evidence at the base of every argument in the assurance case because they contain the information relevant to determining whether the argument's conclusions are adequately supported. Thus, the integrity of the data items is crucial to the adequacy of the case. This importance is emphasized in the confidence arguments, all four of which include the specific assumption DataItemChars **that the data items have the characteristics described in 11.0.a: 1) unambiguous; 2) complete; 3) verifiable; 4) consistent; 5) modifiable; and 6) traceable.**

The possession of these characteristics is explicated as an assumption because the guidance is not consistently clear about how possession must be shown. For some of the contents of some of the data items, the guidance includes specific objectives to demonstrate some of these attributes. For example, high-level requirements must be shown to be unambiguous, consistent, and traceable for Level D and above (see HLRAccCons and HLRTrace2SR in figure 4), and verifiable for Level C (see HLRVerifiable in figure 13). Similarly, low-level requirements must be shown to possess these characteristics for Level C (see LLRAccCons and LLRTrace2HLR in figure 19) and verifiable for Level B (HLRVerifiable in the argument for AddedObjsLevBSat, which was not shown). However, no specific objectives exist for either high- or low-level requirements that directly demand a showing of completeness or modifiability.

The glossary entry for configuration management suggests that it may be responsible for ensuring data item integrity: "the process of … (d) verifying the correctness and completeness of configuration items." However, the specific objectives associated with configuration management do not require any showing of correctness or completeness; therefore, no such conclusion or premise is contained in the assurance case.

The text of DO-178C implies that the integrity of the data items is critically important. The assurance case explicitly emphasizes this importance and highlights the fact that demonstrating the integrity of every aspect of every data item is not necessarily required by the guidance. The extent to which different assessors require such demonstrations may well account for some of the variations in the anecdotally reported amount of effort needed to gain approvals.

### 4.2.4 Observation 4 – Warrants are Difficult to Discern

Throughout the explanation of the assurance case, instances were highlighted for which neither the guidance nor the supporting information provided insight into the reasons for certain objectives. The minimal speculation guiding principle prohibited guesses being made about reasons in these instances. Thus, the warrants that were developed were necessarily trivial. No other option consistent with the principle was possible. Different wording of the trivial warrants could have been employed, but any wording consistent with the text would be equivalent in meaning to the wording that was chosen.

If the guiding principle had been relaxed, the result would likely have been chaotic. That is, every person reading the assurance case would find something with which to disagree. Based on the experiences in SC-205/WG-71 with the attempts to create a comprehensive rationale, some of these disagreements would be quite intense. Although a general consensus existed about the need for nearly all of the objectives, little agreement existed about the specific reasons for them. This lack of agreement about specific reasons was even more pronounced concerning the assignment of objectives to software levels. Therefore, relaxing the minimal speculation principle almost certainly would have derailed the project after the first review by others.

Warrants are difficult, perhaps often impossible, to discern after the fact. The assurance case as written emphasizes this difficulty. This difficulty may also contribute to some of the differences that applicants report noticing among different assessors, perhaps even a stronger contributor than data item integrity.

### 4.2.5 Observation 5 – Adequacy Depends on Specifics

In the abstract, the main arguments, augmented by the associated confidence arguments, seem generally adequate to support their conclusions. Consider the main argument at Level D. Given the associated context and foundational assumption, if it is possible to demonstrate both that high-level requirements are a satisfactory refinement of the allocated system requirements (HLRSatSRRefLevD) and that executable object code is a satisfactory refinement of the high-level requirements (EOCSatHLRefLevD), then it is reasonable to conclude that the software is correct (to an appropriate degree of confidence) with respect to its requirements. Therefore, in the context of an appropriate safety assessment process, it is reasonable to conclude (to an appropriate degree of confidence) that the software performs its intended function at an acceptable level of safety for Level D.

Assessing the adequacy of the supporting arguments in the abstract is difficult. Consider the Level D supporting argument for HLRSatSRRefLevD. It may be asked if it is reasonable to make this conclusion based on showing only compliance with system requirements (HLRComply); accuracy and consistency (HLRAccCons); and traceability to system requirements (HLRTrace2SR). If it is, it may be questioned why the guidance requires additional conclusions to be shown for Level C (HLRVerifiable, HLRConformStd, AlgorAccReq) and another for Level B (HLRCompatTC). The answers lie in a variety of details, which can be known only for an actual project, not in the abstract. Among the details that are needed are the contents of the relevant data items associated with high-level requirements (11.9 Software Requirements Data, 11.14 Software Verification Results); a qualitative understanding of the degree of confidence needed for this conclusion; and the contents of the data items associated with all relevant aspects of the confidence argument for Level D.

The need to have project specifics for assessing argument adequacy suggests that there may well be wisdom in the FAA's long-standing practice of granting software approvals only for specific products. The need also suggests a possible follow-on project: creating an assurance case for a realistic software product.

### 4.2.6 Observation 6 – A Case-Based Alternative Approach Seems Feasible

Moving to the opposite end of the specificity spectrum and looking at the explicated assurance case overall suggests the likely feasibility of developing an entirely case-based alternative set of objectives. This feasibility has already been recognized by the FAA. A key component of the ongoing streamlining assurance processes effort involves creating a minimal set of overarching properties. The Explicate '78 assurance case has contributed to the formulation of the initial draft of the overarching properties. Also, an assurance-case-based approach is currently being followed for exploring the criteria for evaluating the properties. If the effort is successful, the overarching properties will eventually constitute the foundation of a different, optional approval approach.

## 5. THE OTHER DOCUMENTS

Another activity specified to be part of the Explicate '78 research was the following: determining whether there is a need to conduct an analysis similar to the DO-178C analysis for any of the four supplementary documents associated with DO-178C and conducting any such analysis deemed worthwhile.

The four additional documents that were considered were mentioned briefly in section 2.1:

- DO-330: Software Tool Qualification Considerations
- DO-331: Model-Based Development and Verification Supplement to DO-178C and DO-278A
- DO-332: Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A
- DO-333: Formal Methods Supplement to DO-178C and DO-278A

In response to a specific request from the FAA sponsors of this research, the primary effort in this area was expended on creating a full collection of arguments for DO-330. These arguments are presented and explained in appendix C.

For DO-331, DO-332, and DO-333, only the arguments addressing the refining of high-level requirements from allocated system requirements for Level D were created, specifically:

- HLRSatSRRefLevDFM: **High-level requirements are (for Level D using DO-333) a satisfactory refinement of the allocated system requirements** (see figure 32)
- HLRSatSRRefLevDOO: **High-level requirements are (for Level D using DO-332) a satisfactory refinement of the allocated system requirements** (see figure 33)

- HLRSatSRRefLevDFMOO: **High-level requirements are (for Level D using DO-332 and DO-333) a satisfactory refinement of the allocated system requirements** (see figure 34)
- HLRSatSRRefLevDMB: **High-level requirements are (for Level D using DO-331) a satisfactory refinement of the allocated system requirements** (see figure 35)

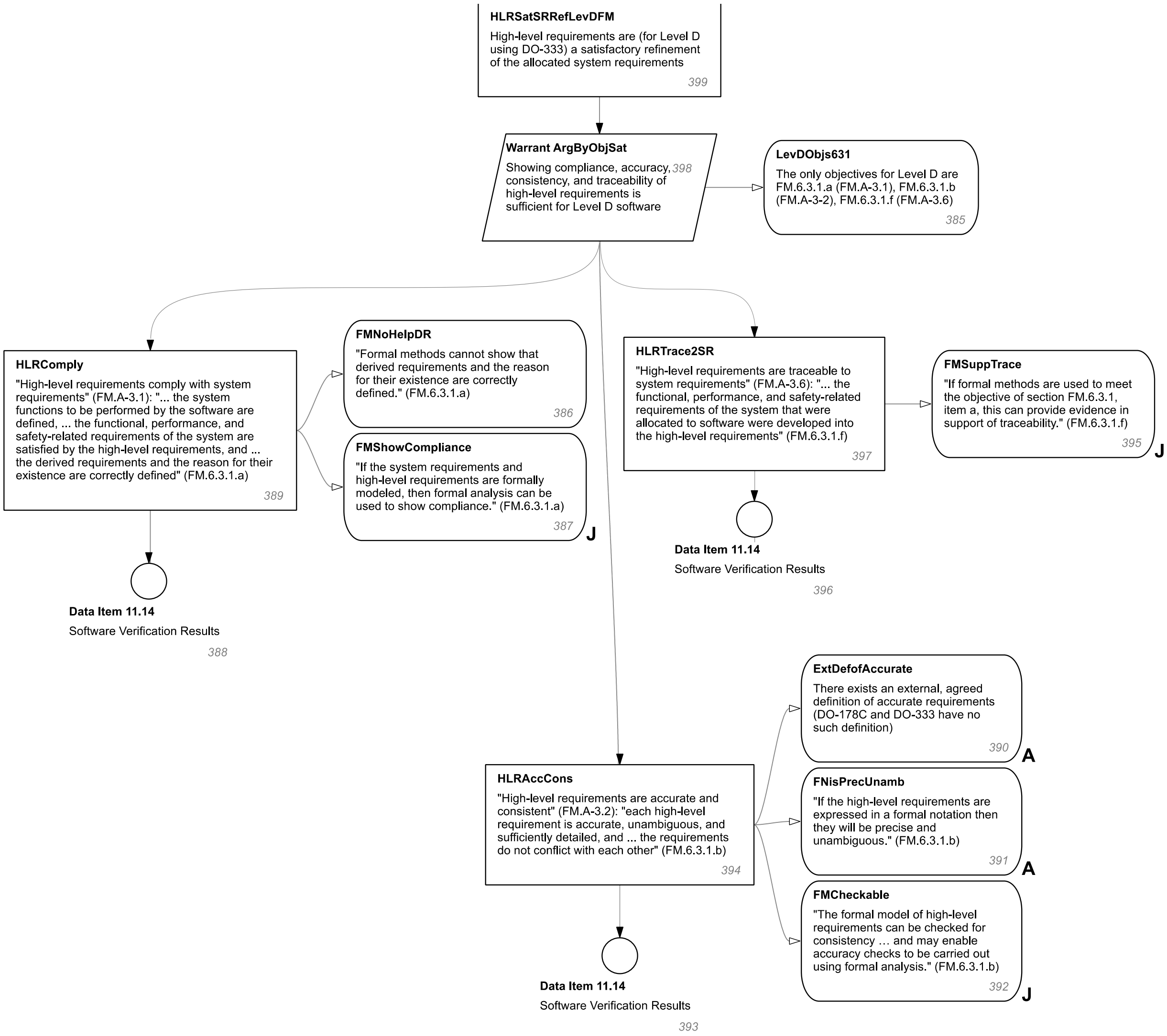

HLRSatSRRefLevDFM
High-level requirements are (for Level D using DO-333) a satisfactory refinement of the allocated system requirements
399
Warrant ArgByObjSat
Showing compliance, accuracy, consistency, and traceability of high-level requirements is sufficient for Level D software
398
LevDObjs631
The only objectives for Level D are FM.6.3.1.a (FM.A-3.1), FM.6.3.1.b (FM.A-3-2), FM.6.3.1.f (FM.A-3.6)
385
HLRComply
"High-level requirements comply with system requirements" (FM.A-3.1): "... the system functions to be performed by the software are defined, ... the functional, performance, and safety-related requirements of the system are satisfied by the high-level requirements, and ... the derived requirements and the reason for their existence are correctly defined" (FM.6.3.1.a)
389
FMNoHelpDR
"Formal methods cannot show that derived requirements and the reason for their existence are correctly defined." (FM.6.3.1.a)
386
FMShowCompliance
"If the system requirements and high-level requirements are formally modeled, then formal analysis can be used to show compliance." (FM.6.3.1.a)
387
J
HLRTrace2SR
"High-level requirements are traceable to system requirements" (FM.A-3.6): "... the functional, performance, and safety-related requirements of the system that were allocated to software were developed into the high-level requirements" (FM.6.3.1.f)
397
FMSuppTrace
"If formal methods are used to meet the objective of section FM.6.3.1, item a, this can provide evidence in support of traceability." (FM.6.3.1.f)
395
J
Data Item 11.14
Software Verification Results
396
Data Item 11.14
Software Verification Results
388
ExtDefofAccurate
There exists an external, agreed definition of accurate requirements (DO-178C and DO-333 have no such definition)
390
A
HLRAccCons
"High-level requirements are accurate and consistent" (FM.A-3.2): "each high-level requirement is accurate, unambiguous, and sufficiently detailed, and ... the requirements do not conflict with each other" (FM.6.3.1.b)
394
FNisPrecUnamb
"If the high-level requirements are expressed in a formal notation then they will be precise and unambiguous." (FM.6.3.1.b)
391
A
FMCheckable
"The formal model of high-level requirements can be checked for consistency … and may enable accuracy checks to be carried out using formal analysis." (FM.6.3.1.b)
392
J
Data Item 11.14
Software Verification Results
393

**Figure 32. Level D (FM): HLRSatSRRefLevDFM**

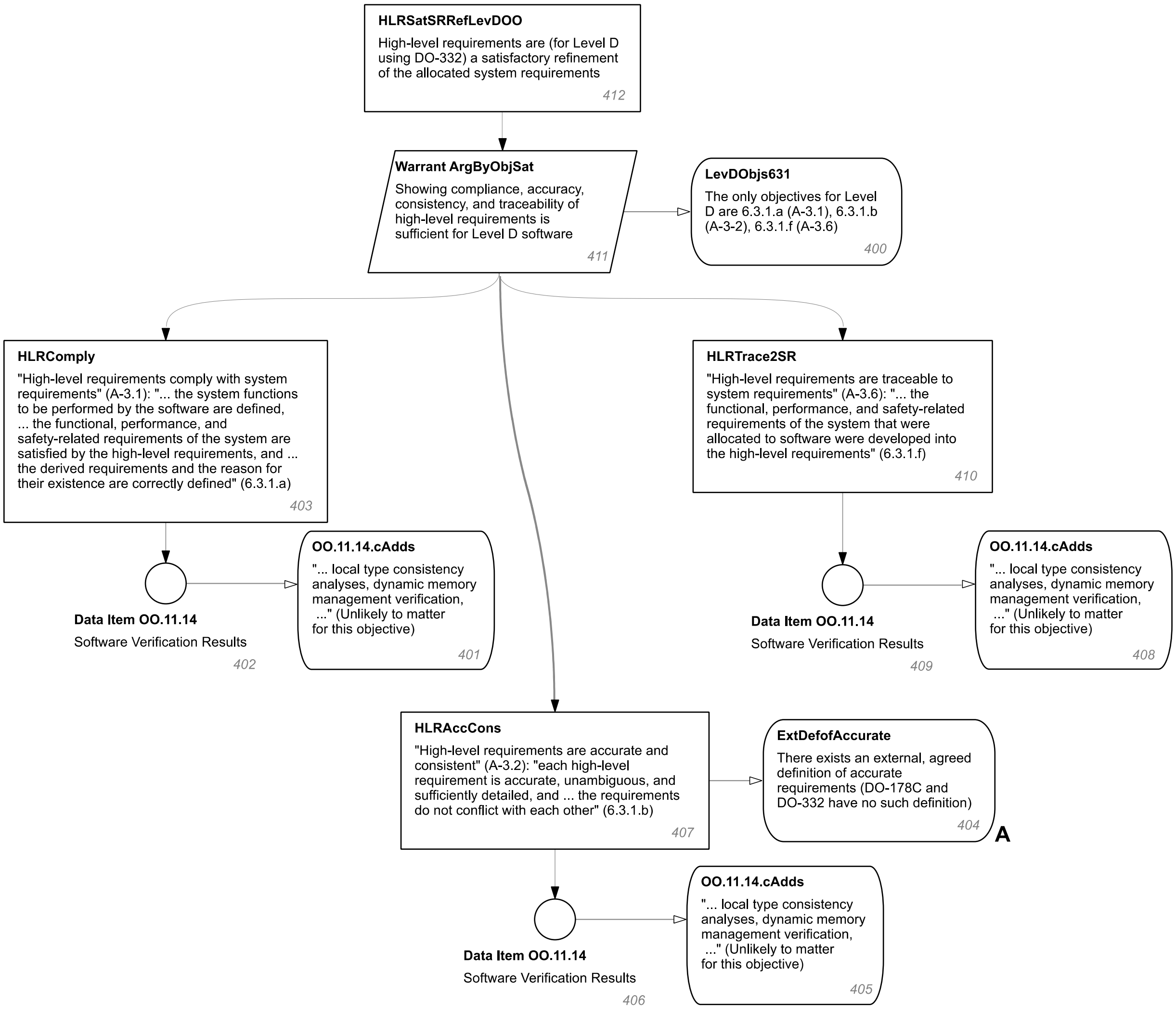
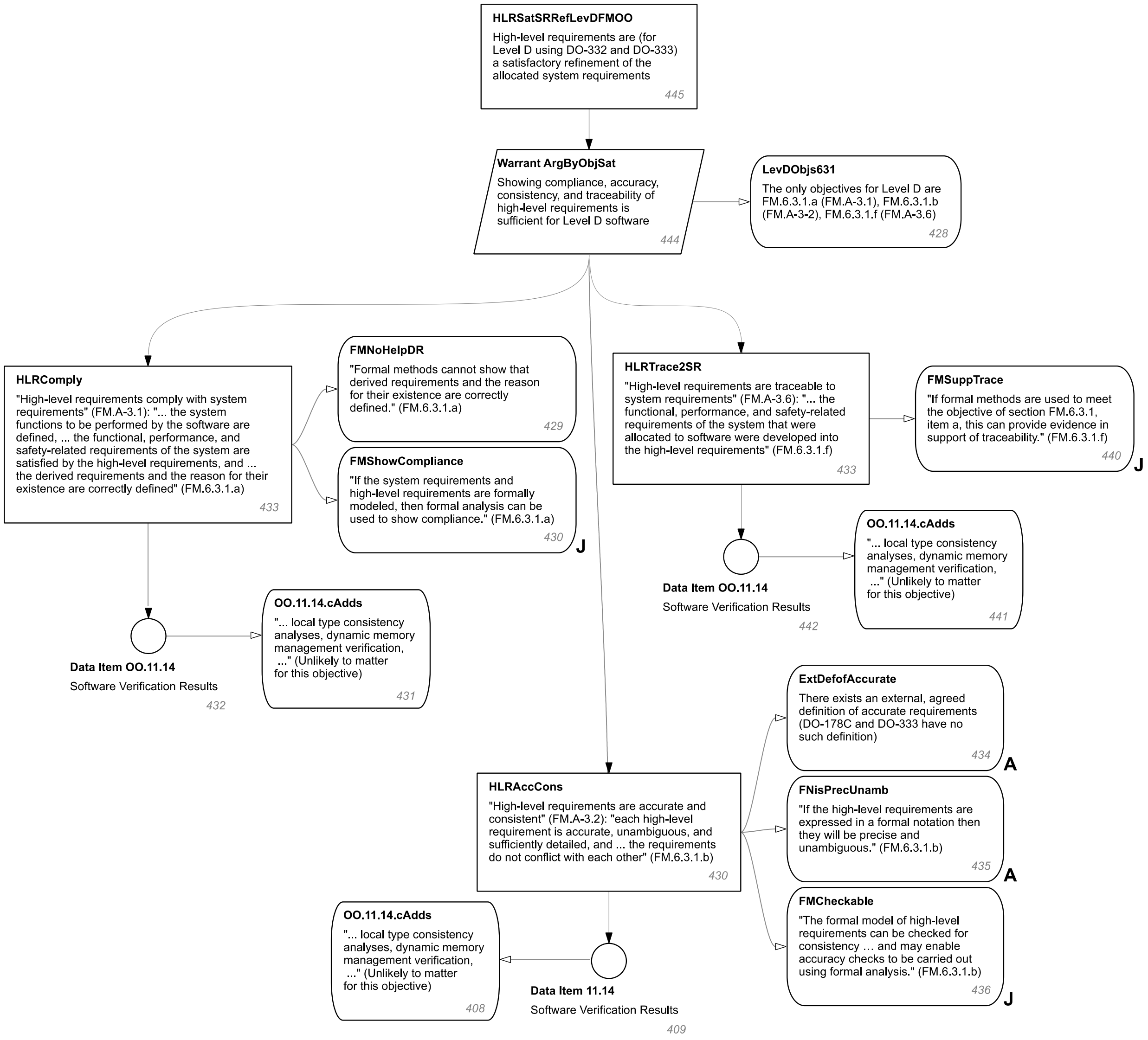

HLRSatSRRefLevDOO
High-level requirements are (for Level D using DO-332) a satisfactory refinement of the allocated system requirements
412
Warrant ArgByObjSat
Showing compliance, accuracy, consistency, and traceability of high-level requirements is sufficient for Level D software
411
LevDObjs631
The only objectives for Level D are 6.3.1.a (A-3.1), 6.3.1.b (A-3-2), 6.3.1.f (A-3.6)
400
HLRComply
"High-level requirements comply with system requirements" (A-3.1): "... the system functions to be performed by the software are defined, ... the functional, performance, and safety-related requirements of the system are satisfied by the high-level requirements, and ... the derived requirements and the reason for their existence are correctly defined" (6.3.1.a)
403
HLRTrace2SR
"High-level requirements are traceable to system requirements" (A-3.6): "... the functional, performance, and safety-related requirements of the system that were allocated to software were developed into the high-level requirements" (6.3.1.f)
410
OO.11.14.cAdds
"... local type consistency analyses, dynamic memory management verification, ..." (Unlikely to matter for this objective)
401
Data Item OO.11.14
Software Verification Results
402
OO.11.14.cAdds
"... local type consistency analyses, dynamic memory management verification, ..." (Unlikely to matter for this objective)
408
Data Item OO.11.14
Software Verification Results
409
HLRAccCons
"High-level requirements are accurate and consistent" (A-3.2): "each high-level requirement is accurate, unambiguous, and sufficiently detailed, and ... the requirements do not conflict with each other" (6.3.1.b)
407
ExtDefofAccurate
There exists an external, agreed definition of accurate requirements (DO-178C and DO-332 have no such definition)
404
A
OO.11.14.cAdds
"... local type consistency analyses, dynamic memory management verification, ..." (Unlikely to matter for this objective)
405
Data Item OO.11.14
Software Verification Results
406

**Figure 33. Level D (OO): HLRSatSRRefLevDOO**


HLRSatSRRefLevDFMOO
High-level requirements are (for Level D using DO-332 and DO-333) a satisfactory refinement of the allocated system requirements
445
Warrant ArgByObjSat
Showing compliance, accuracy, consistency, and traceability of high-level requirements is sufficient for Level D software
444
LevDObjs631
The only objectives for Level D are FM.6.3.1.a (FM.A-3.1), FM.6.3.1.b (FM.A-3-2), FM.6.3.1.f (FM.A-3.6)
428
HLRComply
"High-level requirements comply with system requirements" (FM.A-3.1): "... the system functions to be performed by the software are defined, ... the functional, performance, and safety-related requirements of the system are satisfied by the high-level requirements, and ... the derived requirements and the reason for their existence are correctly defined" (FM.6.3.1.a)
433
FMNoHelpDR
"Formal methods cannot show that derived requirements and the reason for their existence are correctly defined." (FM.6.3.1.a)
429
FMShowCompliance
"If the system requirements and high-level requirements are formally modeled, then formal analysis can be used to show compliance." (FM.6.3.1.a)
430
J
HLRTrace2SR
"High-level requirements are traceable to system requirements" (FM.A-3.6): "... the functional, performance, and safety-related requirements of the system that were allocated to software were developed into the high-level requirements" (FM.6.3.1.f)
433
FMSuppTrace
"If formal methods are used to meet the objective of section FM.6.3.1, item a, this can provide evidence in support of traceability." (FM.6.3.1.f)
440
J
OO.11.14.cAdds
"... local type consistency analyses, dynamic memory management verification, ..." (Unlikely to matter for this objective)
441
Data Item OO.11.14
Software Verification Results
442
Data Item OO.11.14
Software Verification Results
432
OO.11.14.cAdds
"... local type consistency analyses, dynamic memory management verification, ..." (Unlikely to matter for this objective)
431
ExtDefofAccurate
There exists an external, agreed definition of accurate requirements (DO-178C and DO-333 have no such definition)
434
A
HLRAccCons
"High-level requirements are accurate and consistent" (FM.A-3.2): "each high-level requirement is accurate, unambiguous, and sufficiently detailed, and ... the requirements do not conflict with each other" (FM.6.3.1.b)
430
FNisPrecUnamb
"If the high-level requirements are expressed in a formal notation then they will be precise and unambiguous." (FM.6.3.1.b)
435
A
FMCheckable
"The formal model of high-level requirements can be checked for consistency … and may enable accuracy checks to be carried out using formal analysis." (FM.6.3.1.b)
436
J
OO.11.14.cAdds
"... local type consistency analyses, dynamic memory management verification, ..." (Unlikely to matter for this objective)
408
Data Item 11.14
Software Verification Results
409

# Figure 34. Level D (FM & OO): HLRSatSRRefLevDFMOO

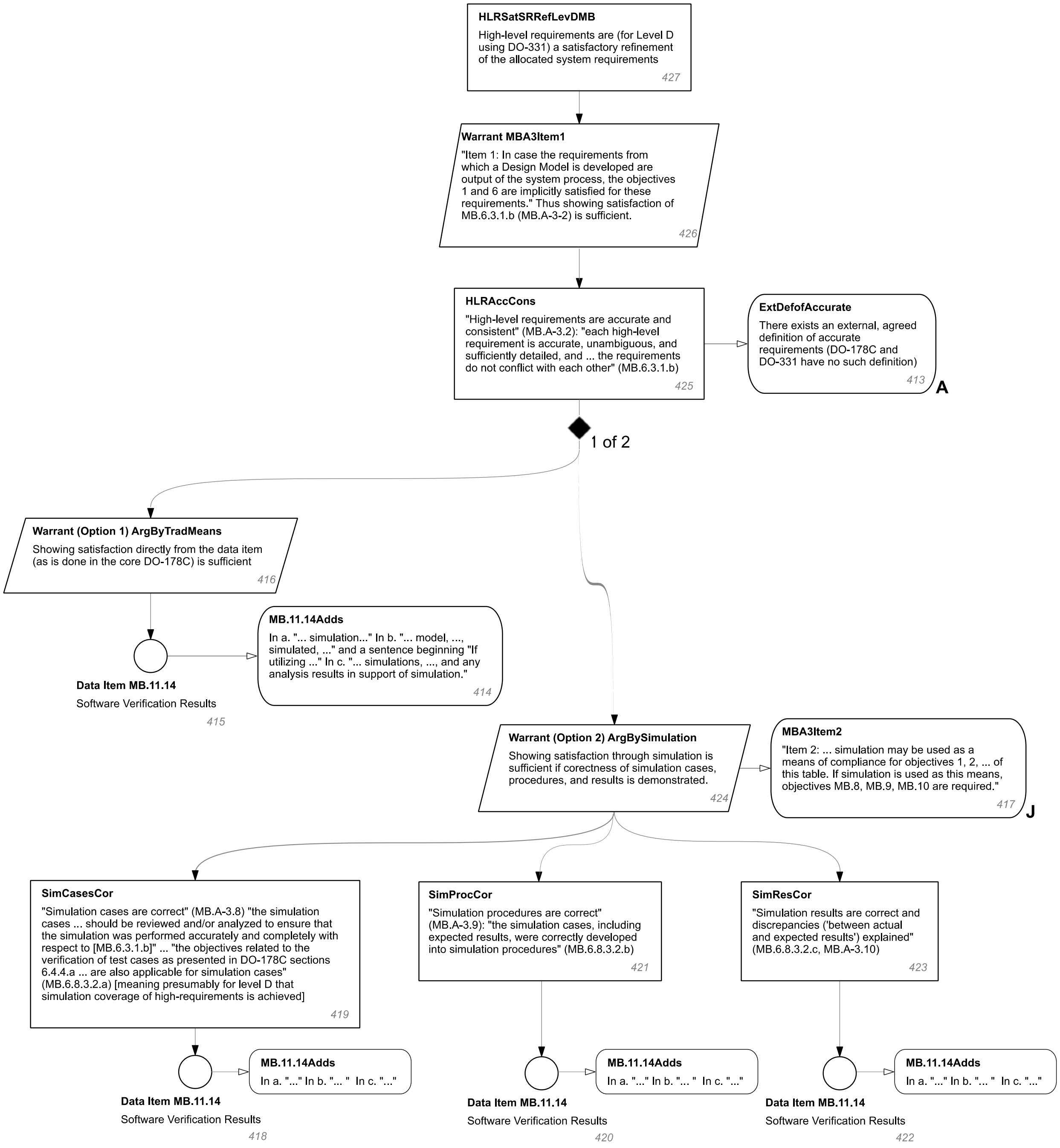

HLRSatSRRefLevDMB
High-level requirements are (for Level D using DO-331) a satisfactory refinement of the allocated system requirements
427
Warrant MBA3Item1
"Item 1: In case the requirements from which a Design Model is developed are output of the system process, the objectives 1 and 6 are implicitly satisfied for these requirements." Thus showing satisfaction of MB.6.3.1.b (MB.A-3-2) is sufficient.
426
HLRAccCons
"High-level requirements are accurate and consistent" (MB.A-3.2): "each high-level requirement is accurate, unambiguous, and sufficiently detailed, and ... the requirements do not conflict with each other" (MB.6.3.1.b)
425
ExtDefofAccurate
There exists an external, agreed definition of accurate requirements (DO-178C and DO-331 have no such definition)
413
A
1 of 2
Warrant (Option 1) ArgByTradMeans
Showing satisfaction directly from the data item (as is done in the core DO-178C) is sufficient
416
MB.11.14Adds
In a. "... simulation..." In b. "... model, ..., simulated, ..." and a sentence beginning "If utilizing ..." In c. "... simulations, ..., and any analysis results in support of simulation."
414
Data Item MB.11.14
Software Verification Results
415
Warrant (Option 2) ArgBySimulation
Showing satisfaction through simulation is sufficient if corectness of simulation cases, procedures, and results is demonstrated.
424
MBA3Item2
"Item 2: ... simulation may be used as a means of compliance for objectives 1, 2, ... of this table. If simulation is used as this means, objectives MB.8, MB.9, MB.10 are required."
417
J
SimCasesCor
"Simulation cases are correct" (MB.A-3.8) "the simulation cases ... should be reviewed and/or analyzed to ensure that the simulation was performed accurately and completely with respect to [MB.6.3.1.b]" ... "the objectives related to the verification of test cases as presented in DO-178C sections 6.4.4.a ... are also applicable for simulation cases" (MB.6.8.3.2.a) [meaning presumably for level D that simulation coverage of high-requirements is achieved]
419
SimProcCor
"Simulation procedures are correct" (MB.A-3.9): "the simulation cases, including expected results, were correctly developed into simulation procedures" (MB.6.8.3.2.b)
421
SimResCor
"Simulation results are correct and discrepancies ('between actual and expected results') explained" (MB.6.8.3.2.c, MB.A-3.10)
423
MB.11.14Adds
In a. "..." In b. "... "  In c. "..."
Data Item MB.11.14
Software Verification Results
418
MB.11.14Adds
In a. "..." In b. "... "  In c. "..."
Data Item MB.11.14
Software Verification Results
420
MB.11.14Adds
In a. "..." In b. "... "  In c. "..."
Data Item MB.11.14
Software Verification Results
422

**Figure 35. Level D (MBD): HLRSatSRRefLevDMB (PARTIAL)**

The argument for HLRSatSRRefLevDFM is identical in form to the argument for the conclusion HLRSatSRRefLevD in the assurance case for Level D for the unsupplemented DO-178C. The differences lie in the addition of an explanatory context element (FMNoHelpDR) and a justification (FMShowCompliance) for HLRComply; an assumption (FNisPrecUnamb) and a justification (FMCheckable) for HLRAccCons; and a justification (FMSuppTrace) for HLRTrace2SR.

The argument for HLRSatSRRefLevDOO is even more similar to the argument for the conclusion HLRSatSRRefLevD. The only difference lies in the inclusion of a context element describing the addition made in DO-332 to the Software Verification Results data item. As noted, this addition is not likely to be relevant in establishing the truth of the three premises relevant to this argument.

The argument for HLRSatSRRefLevDOO combines the content of the two previous arguments. Because no overlap exists in the additional content from DO-333 and DO-332, the two are easy to combine.

A similar comment cannot be made concerning using DO-331 with either (or both) of the other two supplements. The argument presented for HLRSatSRRefLevDMB is only a partial argument, considering only one of the three premises (HLRAccCons). The diagram introduces another bit of notation: The black diamond denotes possible alternatives; the 1 of 2 text denotes that one and only one of the two alternatives contributes to supporting the conclusion. HLRAccCons is supported by one of the two warrants. Warrant ArgByTradMeans indicates **showing satisfaction directly from the data item (as is done in the core DO-178C).** Warrant ArgBySimulation indicates showing satisfaction through simulation, which requires demonstrating SimCasesCor, SimProcCor, and SimResCor. The evidence for these must be included in the Software Verification Results; the additional information that DO-331 requires be included in this data item is indicated by the context element MB.11.14Adds.

## 6. CONCLUDING REMARKS

This report has documented two of the main achievements of the Explicate '78 research:

1. Expressing as an assurance case the arguments contained in, or implied by, DO-178C, which implicitly justify the assumption that the document meets its stated purpose of providing "guidelines 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"
2. Expressing as an assurance case the arguments contained in, or implied by, DO-330, whose stated purpose "is to provide tool qualification guidance"

Substantial portions of the DO-178C assurance case were presented and explained in the body of the report, with the entire case presented in appendix A. The complete DO-330 assurance case was presented and explained in appendix B. Brief, but substantive, explanatory material associated with DO-178C and with assurance cases was also presented in the body of the report.

Completion of this report brings to an end the current Explicate '78 research. The spirit of the research continues, however, in the ongoing effort to define, refine, and put into practice a new approach to assurance based on overarching properties.

## 7. REFERENCES

1. RTCA. (2011). DO-178C: Software Considerations in Airborne Systems and Equipment Certification, RTCA, Inc., Washington, DC. (Also published as EUROCAE ED-12C.)

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

3. Holloway, C. M. Understanding assurance cases: Module 3 - Evaluation. Developed for the FAA under Annex 2 of IAI-1073, September 2015.

4. RTCA. (2011). DO-333: Formal Methods Supplement to DO-178C and DO-278A, RTCA, Inc., Washington, DC. (Also published as EUROCAE ED-216.)

5. Holloway, C. M. Understanding assurance cases: Module 4 - Creation. Developed for the FAA under Annex 2 of IAI-1073, March 2016.

6. RTCA. (2011). DO-178A: Software integrity assurance considerations for communication, navigation, surveillance, and air traffic management (CNS/ATM) systems, RTCA, Inc., Washington, DC. (Also published as EUROCAE ED-109A.)

7. Ministry of Defence. (2015. Defence Standard 00-56, Issue 6: Safety management requirements for defense systems — Part 1: Requirements and guidance, Glasgow, UK.

8. NASA Report. (2015). Understanding and evaluating assurance cases. (NASA/CR-2015-218802).

9. RTCA. (2011). DO-331: Model-Based Development and Verification Supplement to DO-178C and DO-278A, RTCA, Inc., Washington, DC. (Also published as EUROCAE ED-218.)

10. RTCA. (2011). DO-330: Software tool qualification considerations, RTCA, Inc., Washington, DC. (Also published as EUROCAE ED-215.)

11. RTCA. (2011). Supporting information for DO-178C and DO-278A. DO-248C (EUROCAE ED-94C). Graydon, P. J. (2014). *Towards a Clearer Understanding of Context and its Role in Assurance Argument Confidence*. Proceedings from the International Conference on Computer Safety, Reliability and Security (SAFECOMP), Florence, Italy.

12. RTCA. (2011). DO-332: Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A, RTCA, Inc., Washington, DC. (Also published as EUROCAE ED-217.)

13. Graydon, P. J., Holloway, C. M. (2017). An Investigation of Proposed Techniques for Quantifying Confidence in Assurance Arguments. *Safety Science,* 92(February 2017), 53–65.

14. GSN Committee. (2011). GSN Community Standard, Version 1, Origin Consulting Ltd., York, UK.

15. Hawkins, R., Kelly, T., Knight, J., and Graydon, P. (2011). *A New Approach to Creating Clear Safety Arguments*. Proceedings from Advances in Systems Safety: Proceedings of the 19th Safety-Critical Systems Symposium, Southampton, UK.

16. Copi, I. M., Cohen, C., and McMahon, K. (2011). *Introduction to Logic,* (14th ed.). Upper Saddle River, NJ: Pearson Education.

17. European Aviation Safety Agency Report. (2013). Software Considerations for Certification of Airborne Systems and Equipment. (AMC 20-115C).

18. FAA. Standard airworthiness certification: Regulations – Title 14 code of federal regulations. https://www.faa.gov/aircraft/air_cert/airworthiness_certification/std_awcert/std_awcert_regs/regs/ (last visited 7 November 2016).

19. FAA. (2013). Advisory circular 20-115C. Airborne Software Assurance. D.C.: Government Publishing Office.

20. Software Engineering Institute. (2015). *Eliminative Argumentation: A Basis for Arguing Confidence in System Properties.* (CMU-SEI-2015-TR-005). Pittsburgh, PA: Goodenough, J. B., Weinstock, C. B., and Klein, A. Z.

21. Toulmin, S. E. (2003). *The Uses of Argument, updated edition*. Cambridge, UK: Cambridge University Press.

22. Kelly, T. P. (2007). *Reviewing Assurance Arguments: A Step-by-Step Approach*. Proceedings from the Workshop on Assurance Cases for Security — The Metrics Challenge (co-located with the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks [DSN]), Edinburgh, UK.

23. Holloway, C. M. Understanding assurance cases: Module 5 - Speculation. Developed for the FAA under Annex 2 of IAI-1073, March 2016.

24. Govier, T. (2010). *A Practical Study of Argument (7th ed.).* Belmont, CA: Cengage Learning.

25. Holloway, C. M. Understanding assurance cases: Module 2 - Application. Developed for the FAA under Annex 2 of IAI-1073, September 2015.

26. Holloway, C. M. (2015). *Explicate '78: Uncovering the implicit assurance case in DO-178C.* Proceedings from Engineering Systems for Safety: Proceedings of the 23rd Safety-critical Systems Symposium, Bristol, UK. M. Parsons and T. Anderson, Eds., Safety Critical Systems Club, pp. 205–225.

27. Holloway, C. M. Understanding Assurance Cases: Module 1 - Foundation. Developed for the FAA under Annex 2 of IAI-1073, September 2015.

28. Ankrum, T. S., and Kromholz, A. H. (2005). *Structured assurance cases: Three common standards.* Proceedings from the Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE), Heidelberg, Germany.

29. Holloway, C. M. (2012). *Towards understanding the DO-178C / ED-12C assurance case.* Proceedings from the 7th IET International Conference on System Safety, Incorporating the Cyber Security Conference, Edinburgh, UK.

30. Graydon, P. J. (2015). *Formal assurance arguments: A solution in search of a problem?* Proceedings from the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil.

31. Holloway, C. M. (2013). *Making the Implicit Explicit: Towards an Assurance Case for DO-178C.* Proceedings from the 31st International System Safety Conference (ISSC), Boston, Massachusetts.

32. Graydon, P., Knight, J., and Green, M. (2010). *Certification and safety cases.* Proceedings from the 28th International Systems Safety Conference (ISSC), Minneapolis, Minnesota.

33. NASA Report. (2015). Current Practices in Constructing and Evaluating Assurance Cases with Applications to Aviation. (NASA/CR-2015-218678).

# APPENDIX A—ARGUMENTS FOR DO-178C

This appendix contains all of the argument diagrams developed for DO-178C.

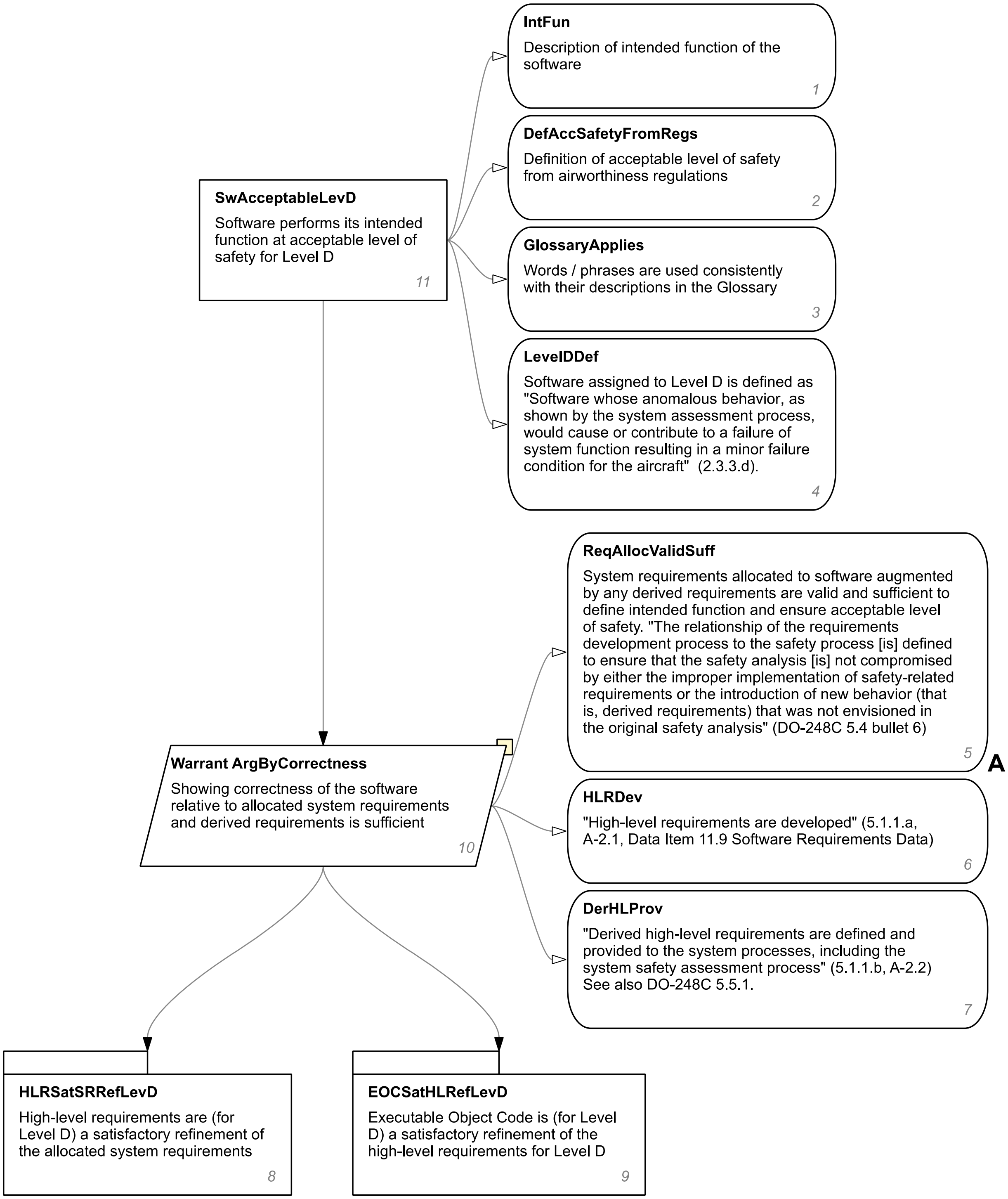

IntFun
Description of intended function of the software
1
DefAccSafetyFromRegs
Definition of acceptable level of safety from airworthiness regulations
2
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
3
LevelDDef
Software assigned to Level D is defined as "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" (2.3.3.d).
4
SwAcceptableLevD
Software performs its intended function at acceptable level of safety for Level D
11
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. "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 original safety analysis" (DO-248C 5.4 bullet 6)
5
A
Warrant ArgByCorrectness
Showing correctness of the software relative to allocated system requirements and derived requirements is sufficient
10
HLRDev
"High-level requirements are developed" (5.1.1.a, A-2.1, Data Item 11.9 Software Requirements Data)
6
DerHLProv
"Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process" (5.1.1.b, A-2.2) See also DO-248C 5.5.1.
7
HLRSatSRRefLevD
High-level requirements are (for Level D) a satisfactory refinement of the allocated system requirements
8
EOCSatHLRefLevD
Executable Object Code is (for Level D) a satisfactory refinement of the high-level requirements for Level D
9

**Figure A-1. Level D: SWACCEPTABLELEVD**

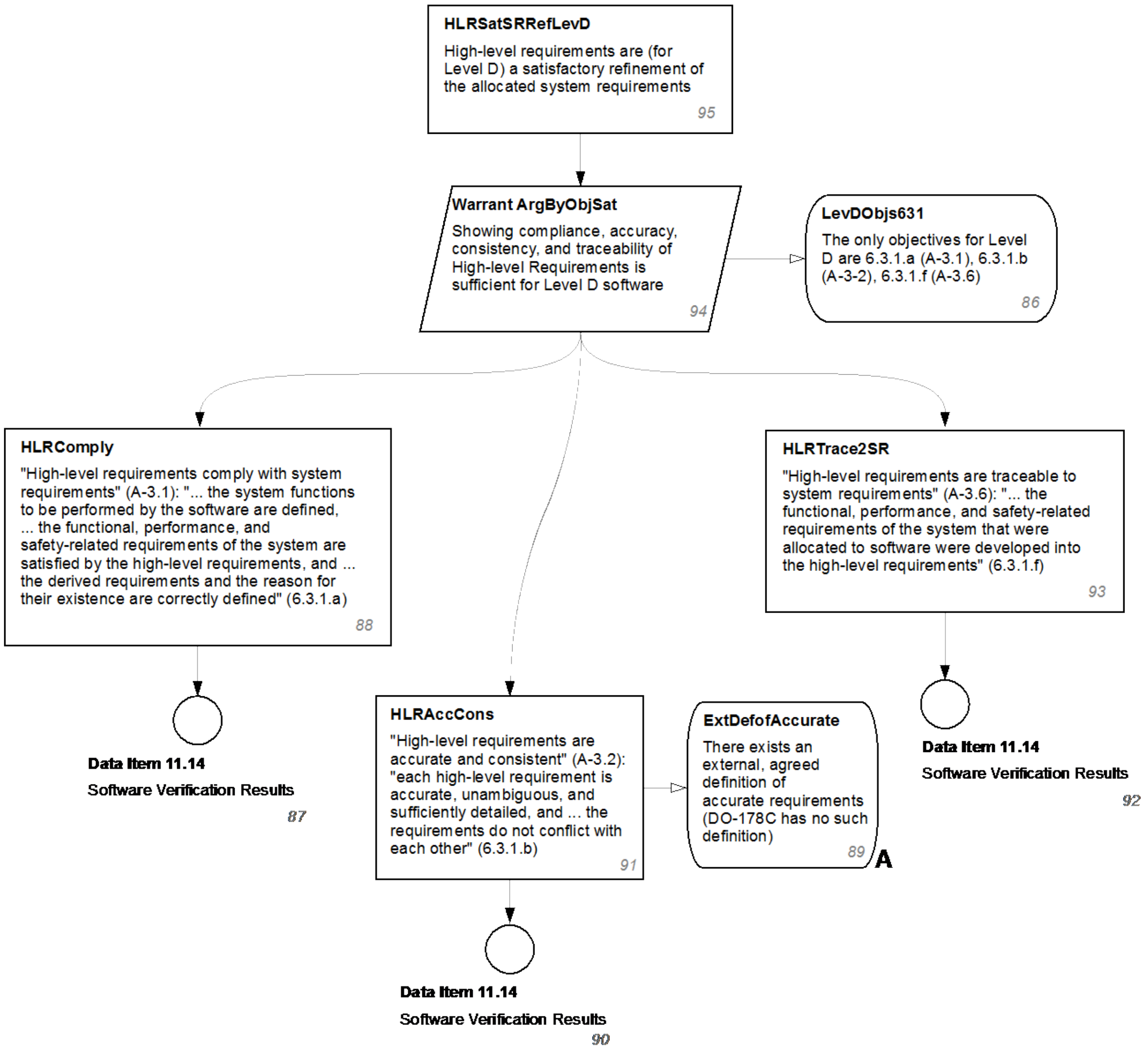

HLRSatSRRefLevD
High-level requirements are (for Level D) a satisfactory refinement of the allocated system requirements
95
Warrant ArgByObjSat
Showing compliance, accuracy, consistency, and traceability of High-level Requirements is sufficient for Level D software
94
LevDObjs631
The only objectives for Level D are 6.3.1.a (A-3.1), 6.3.1.b (A-3-2), 6.3.1.f (A-3.6)
86
HLRComply
"High-level requirements comply with system requirements" (A-3.1): "... the system functions to be performed by the software are defined, ... the functional, performance, and safety-related requirements of the system are satisfied by the high-level requirements, and ... the derived requirements and the reason for their existence are correctly defined" (6.3.1.a)
88
HLRTrace2SR
"High-level requirements are traceable to system requirements" (A-3.6): "... the functional, performance, and safety-related requirements of the system that were allocated to software were developed into the high-level requirements" (6.3.1.f)
93
Data Item 11.14
Software Verification Results
87
HLRAccCons
"High-level requirements are accurate and consistent" (A-3.2): "each high-level requirement is accurate, unambiguous, and sufficiently detailed, and ... the requirements do not conflict with each other" (6.3.1.b)
91
ExtDefofAccurate
There exists an external, agreed definition of accurate requirements (DO-178C has no such definition)
89
A
Data Item 11.14
Software Verification Results
92
Data Item 11.14
Software Verification Results
90

**Figure A-2. Level D: HLRSatSRRefLevD**

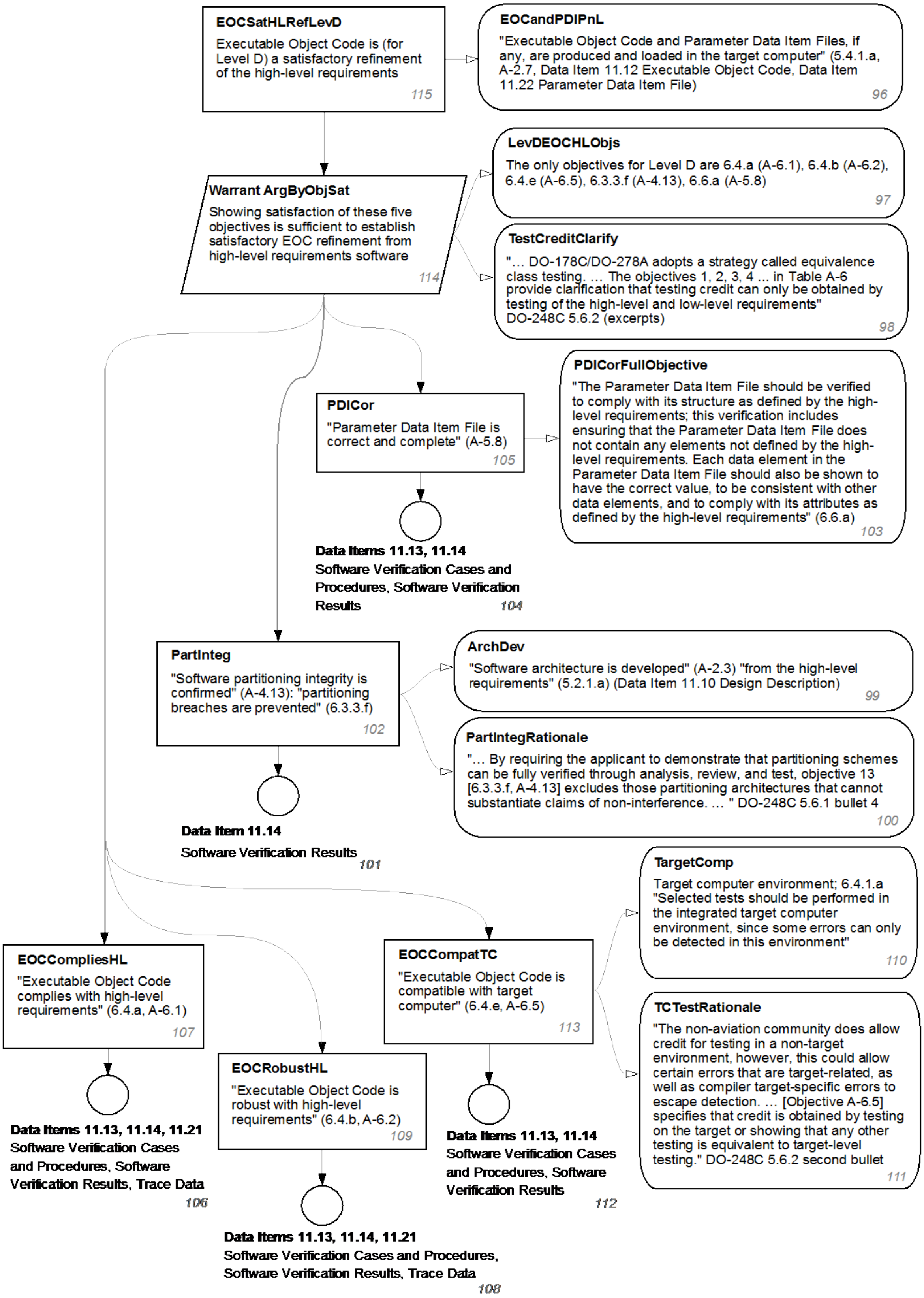

EOCSatHLRefLevD
Executable Object Code is (for Level D) a satisfactory refinement of the high-level requirements
115
EOCandPDIPnL
"Executable Object Code and Parameter Data Item Files, if any, are produced and loaded in the target computer" (5.4.1.a, A-2.7, Data Item 11.12 Executable Object Code, Data Item 11.22 Parameter Data Item File)
96
Warrant ArgByObjSat
Showing satisfaction of these five objectives is sufficient to establish satisfactory EOC refinement from high-level requirements software
114
LevDEOCHLObjs
The only objectives for Level D are 6.4.a (A-6.1), 6.4.b (A-6.2), 6.4.e (A-6.5), 6.3.3.f (A-4.13), 6.6.a (A-5.8)
97
TestCreditClarify
"… DO-178C/DO-278A adopts a strategy called equivalence class testing. … The objectives 1, 2, 3, 4 … in Table A-6 provide clarification that testing credit can only be obtained by testing of the high-level and low-level requirements" DO-248C 5.6.2 (excerpts)
98
PDICorFullObjective
"The Parameter Data Item File should be verified to comply with its structure as defined by the high-level requirements; this verification includes ensuring that the Parameter Data Item File does not contain any elements not defined by the high-level requirements. Each data element in the Parameter Data Item File should also be shown to have the correct value, to be consistent with other data elements, and to comply with its attributes as defined by the high-level requirements" (6.6.a)
103
PDICor
"Parameter Data Item File is correct and complete" (A-5.8)
105
Data Items 11.13, 11.14
Software Verification Cases and Procedures, Software Verification Results
104
PartInteg
"Software partitioning integrity is confirmed" (A-4.13): "partitioning breaches are prevented" (6.3.3.f)
102
ArchDev
"Software architecture is developed" (A-2.3) "from the high-level requirements" (5.2.1.a) (Data Item 11.10 Design Description)
99
PartIntegRationale
"… By requiring the applicant to demonstrate that partitioning schemes can be fully verified through analysis, review, and test, objective 13 [6.3.3.f, A-4.13] excludes those partitioning architectures that cannot substantiate claims of non-interference. … " DO-248C 5.6.1 bullet 4
100
Data Item 11.14
Software Verification Results
101
TargetComp
Target computer environment; 6.4.1.a "Selected tests should be performed in the integrated target computer environment, since some errors can only be detected in this environment"
110
EOCCompliesHL
"Executable Object Code complies with high-level requirements" (6.4.a, A-6.1)
107
EOCCompatTC
"Executable Object Code is compatible with target computer" (6.4.e, A-6.5)
113
TCTestRationale
"The non-aviation community does allow credit for testing in a non-target environment, however, this could allow certain errors that are target-related, as well as compiler target-specific errors to escape detection. … [Objective A-6.5] specifies that credit is obtained by testing on the target or showing that any other testing is equivalent to target-level testing." DO-248C 5.6.2 second bullet
111
EOCRobustHL
"Executable Object Code is robust with high-level requirements" (6.4.b, A-6.2)
109
Data Items 11.13, 11.14, 11.21
Software Verification Cases and Procedures, Software Verification Results, Trace Data
106
Data Items 11.13, 11.14
Software Verification Cases and Procedures, Software Verification Results
112
Data Items 11.13, 11.14, 11.21
Software Verification Cases and Procedures, Software Verification Results, Trace Data
108

**Figure A-3. Level D: EOCSatHLRefLev**

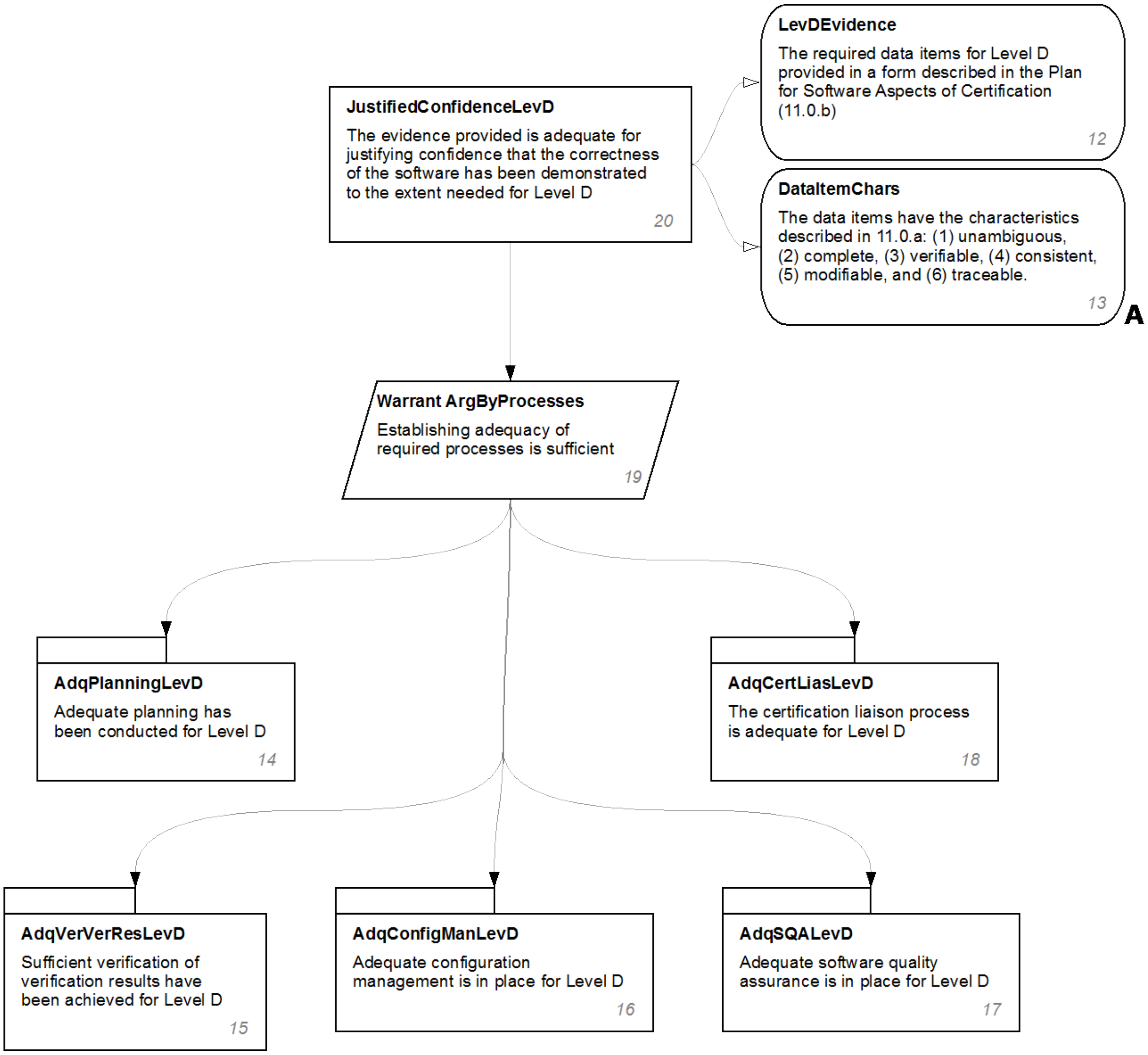

JustifiedConfidenceLevD
The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for Level D
20
LevDEvidence
The required data items for Level D provided in a form described in the Plan for Software Aspects of Certification (11.0.b)
12
DataItemChars
The data items have the characteristics described in 11.0.a: (1) unambiguous, (2) complete, (3) verifiable, (4) consistent, (5) modifiable, and (6) traceable.
13
A
Warrant ArgByProcesses
Establishing adequacy of required processes is sufficient
19
AdqPlanningLevD
Adequate planning has been conducted for Level D
14
AdqCertLiasLevD
The certification liaison process is adequate for Level D
18
AdqVerVerResLevD
Sufficient verification of verification results have been achieved for Level D
15
AdqConfigManLevD
Adequate configuration management is in place for Level D
16
AdqSQALevD
Adequate software quality assurance is in place for Level D
17

**Figure A-4. Level D: JustifiedConfidenceLevD**

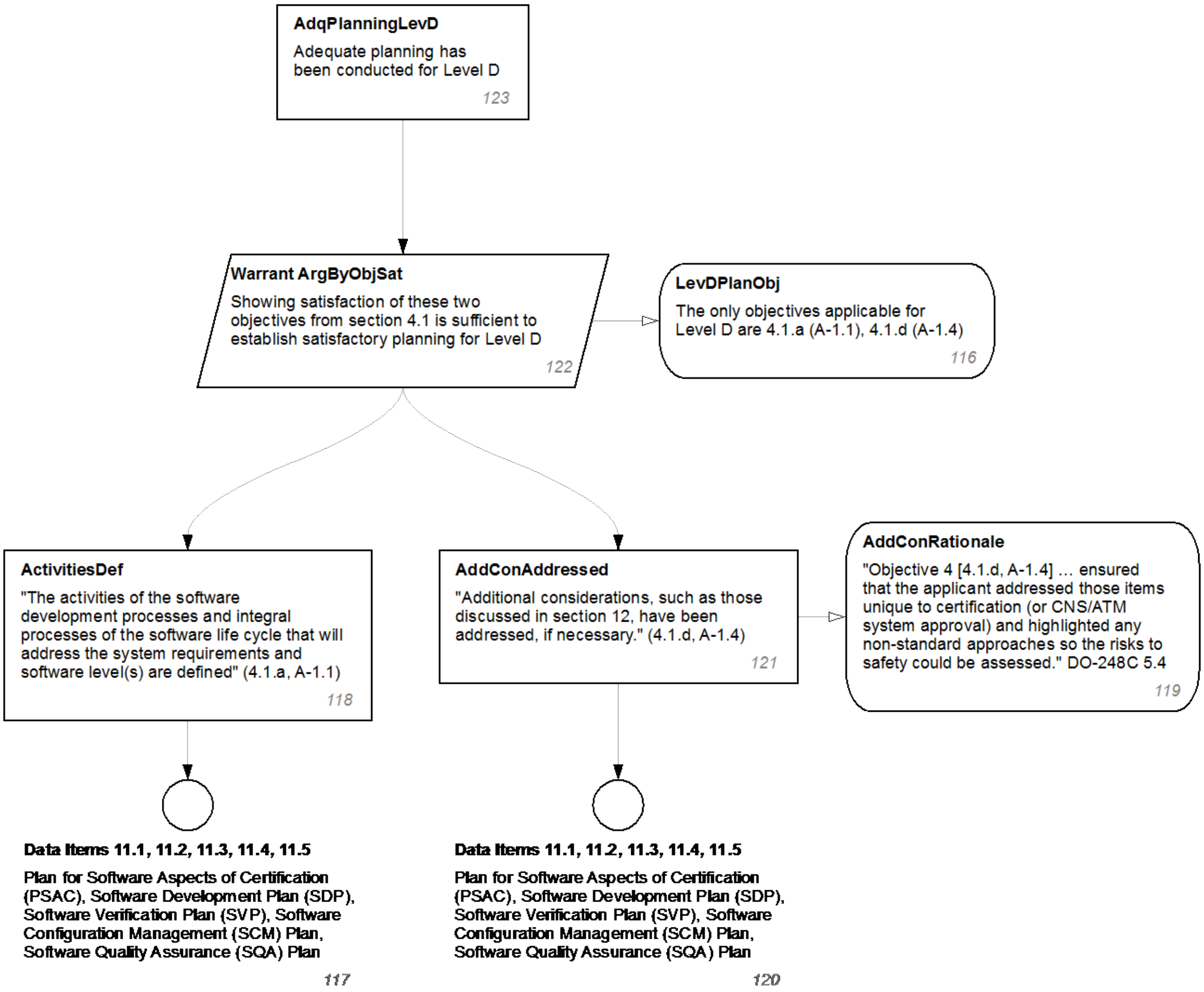

AdqPlanningLevD
Adequate planning has been conducted for Level D
123
Warrant ArgByObjSat
Showing satisfaction of these two objectives from section 4.1 is sufficient to establish satisfactory planning for Level D
122
LevDPlanObj
The only objectives applicable for Level D are 4.1.a (A-1.1), 4.1.d (A-1.4)
116
ActivitiesDef
"The activities of the software development processes and integral processes of the software life cycle that will address the system requirements and software level(s) are defined" (4.1.a, A-1.1)
118
AddConAddressed
"Additional considerations, such as those discussed in section 12, have been addressed, if necessary." (4.1.d, A-1.4)
121
AddConRationale
"Objective 4 [4.1.d, A-1.4] ... ensured that the applicant addressed those items unique to certification (or CNS/ATM system approval) and highlighted any non-standard approaches so the risks to safety could be assessed." DO-248C 5.4
119
Data Items 11.1, 11.2, 11.3, 11.4, 11.5
Plan for Software Aspects of Certification (PSAC), Software Development Plan (SDP), Software Verification Plan (SVP), Software Configuration Management (SCM) Plan, Software Quality Assurance (SQA) Plan
117
Data Items 11.1, 11.2, 11.3, 11.4, 11.5
Plan for Software Aspects of Certification (PSAC), Software Development Plan (SDP), Software Verification Plan (SVP), Software Configuration Management (SCM) Plan, Software Quality Assurance (SQA) Plan
120

**Figure A-5. Level D: ADQPLANNINGLEVD**

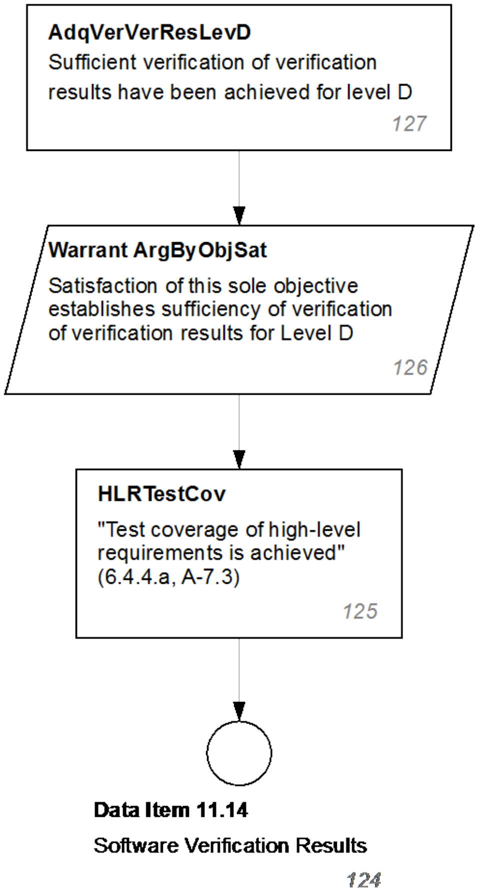

AdqVerVerResLevD
Sufficient verification of verification results have been achieved for level D
127
Warrant ArgByObjSat
Satisfaction of this sole objective establishes sufficiency of verification of verification results for Level D
126
HLRTestCov
"Test coverage of high-level requirements is achieved" (6.4.4.a, A-7.3)
125
Data Item 11.14
Software Verification Results
124

**Figure A-6. Level D: ADQVERVERLEVD**

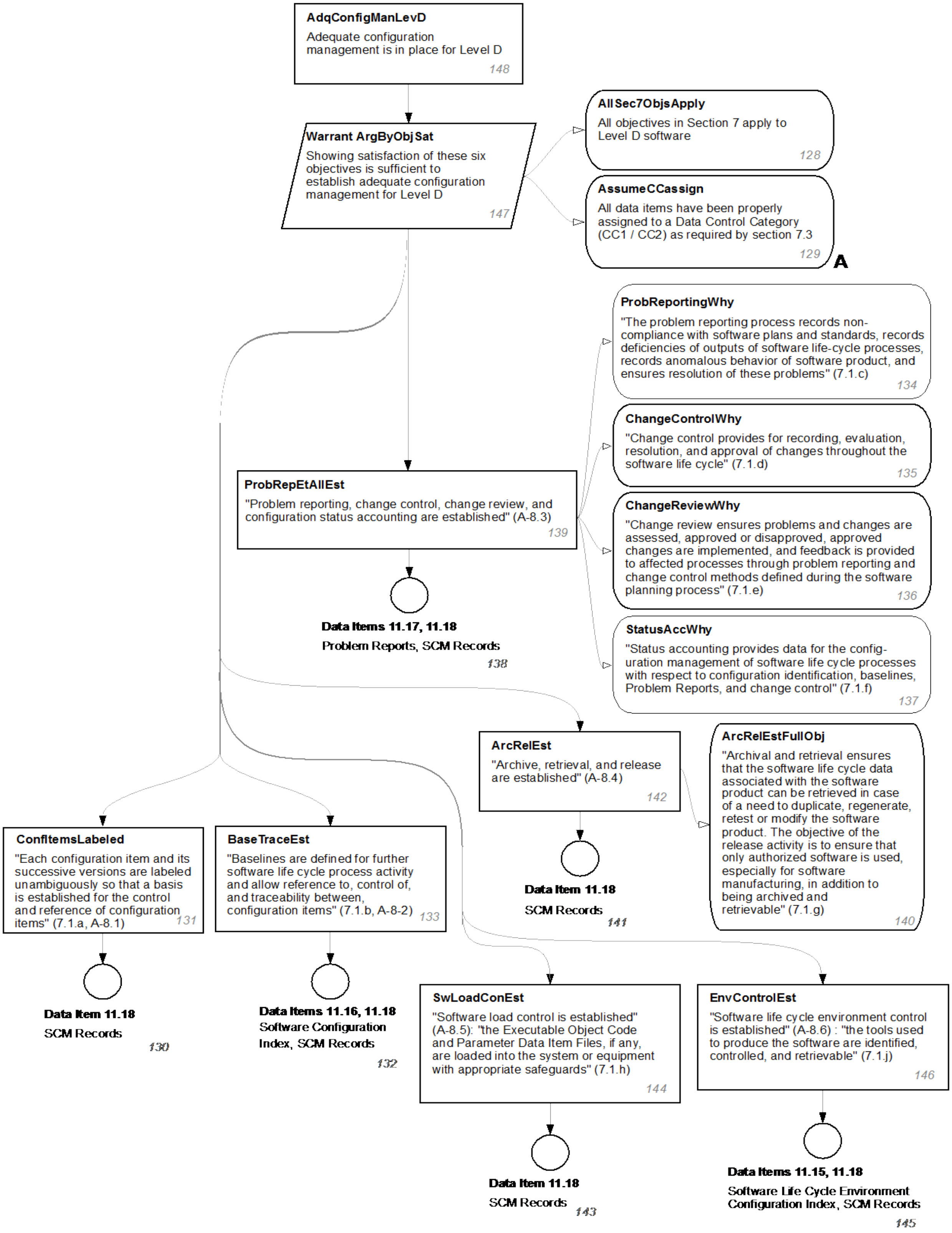

AdqConfigManLevD
Adequate configuration management is in place for Level D
148
Warrant ArgByObjSat
Showing satisfaction of these six objectives is sufficient to establish adequate configuration management for Level D
147
AllSec7ObjsApply
All objectives in Section 7 apply to Level D software
128
AssumeCCassign
All data items have been properly assigned to a Data Control Category (CC1 / CC2) as required by section 7.3
129
A
ProbReportingWhy
"The problem reporting process records non-compliance with software plans and standards, records deficiencies of outputs of software life-cycle processes, records anomalous behavior of software product, and ensures resolution of these problems" (7.1.c)
134
ChangeControlWhy
"Change control provides for recording, evaluation, resolution, and approval of changes throughout the software life cycle" (7.1.d)
135
ProbRepEtAllEst
"Problem reporting, change control, change review, and configuration status accounting are established" (A-8.3)
139
ChangeReviewWhy
"Change review ensures problems and changes are assessed, approved or disapproved, approved changes are implemented, and feedback is provided to affected processes through problem reporting and change control methods defined during the software planning process" (7.1.e)
136
Data Items 11.17, 11.18
Problem Reports, SCM Records
138
StatusAccWhy
"Status accounting provides data for the configuration management of software life cycle processes with respect to configuration identification, baselines, Problem Reports, and change control" (7.1.f)
137
ArcRelEst
"Archive, retrieval, and release are established" (A-8.4)
142
ArcRelEstFullObj
"Archival and retrieval ensures that the software life cycle data associated with the software product can be retrieved in case of a need to duplicate, regenerate, retest or modify the software product. The objective of the release activity is to ensure that only authorized software is used, especially for software manufacturing, in addition to being archived and retrievable" (7.1.g)
140
Data Item 11.18
SCM Records
141
ConfItemsLabeled
"Each configuration item and its successive versions are labeled unambiguously so that a basis is established for the control and reference of configuration items" (7.1.a, A-8.1)
131
BaseTraceEst
"Baselines are defined for further software life cycle process activity and allow reference to, control of, and traceability between, configuration items" (7.1.b, A-8-2)
133
SwLoadConEst
"Software load control is established" (A-8.5): "the Executable Object Code and Parameter Data Item Files, if any, are loaded into the system or equipment with appropriate safeguards" (7.1.h)
144
EnvControlEst
"Software life cycle environment control is established" (A-8.6) : "the tools used to produce the software are identified, controlled, and retrievable" (7.1.j)
146
Data Item 11.18
SCM Records
130
Data Items 11.16, 11.18
Software Configuration Index, SCM Records
132
Data Item 11.18
SCM Records
143
Data Items 11.15, 11.18
Software Life Cycle Environment Configuration Index, SCM Records
145

**Figure A-7. Level D: ADQCONFIGMANLEVD**

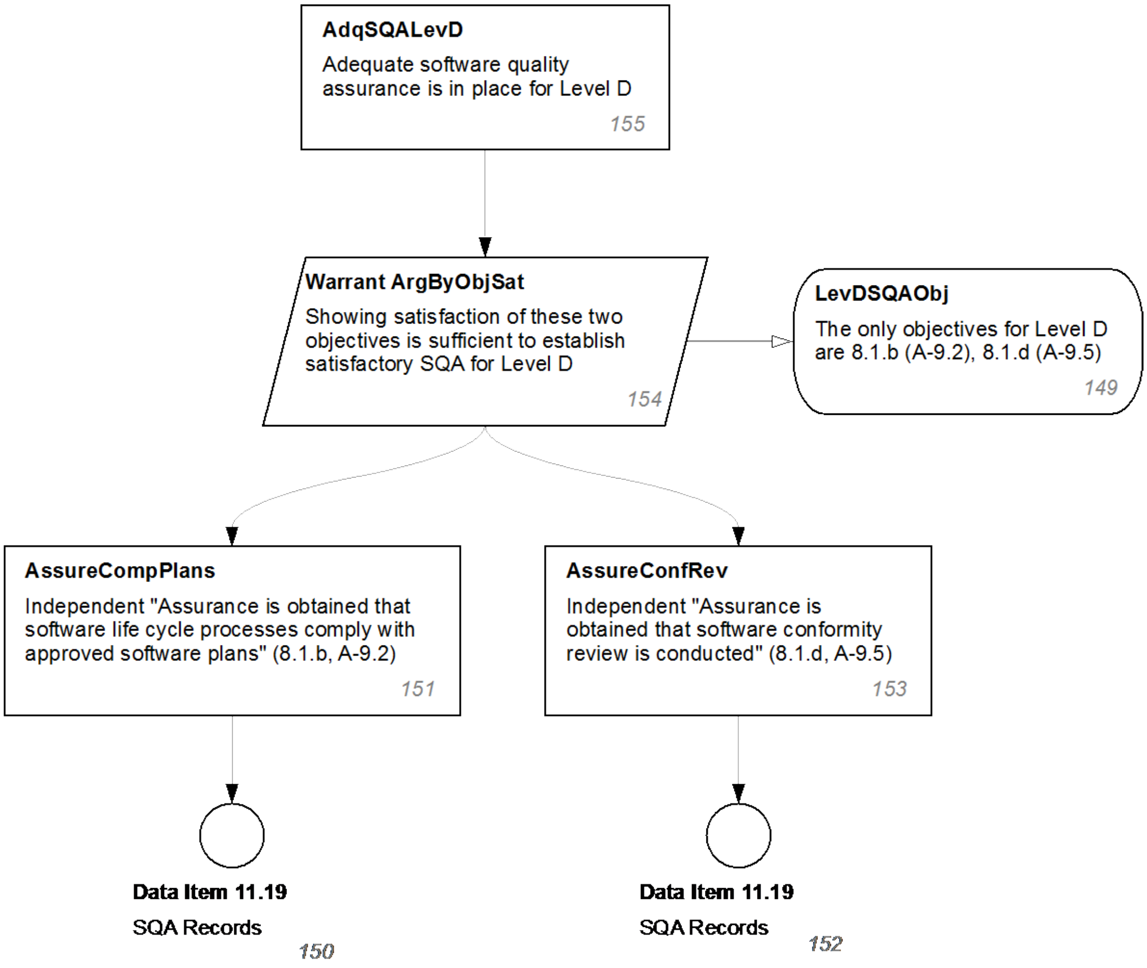

AdqSQALevD
Adequate software quality assurance is in place for Level D
155
Warrant ArgByObjSat
Showing satisfaction of these two objectives is sufficient to establish satisfactory SQA for Level D
154
LevDSQAObj
The only objectives for Level D are 8.1.b (A-9.2), 8.1.d (A-9.5)
149
AssureCompPlans
Independent "Assurance is obtained that software life cycle processes comply with approved software plans" (8.1.b, A-9.2)
151
AssureConfRev
Independent "Assurance is obtained that software conformity review is conducted" (8.1.d, A-9.5)
153
Data Item 11.19
SQA Records
150
Data Item 11.19
SQA Records
152

**Figure A-8. Level D: ADQSQALEVD**

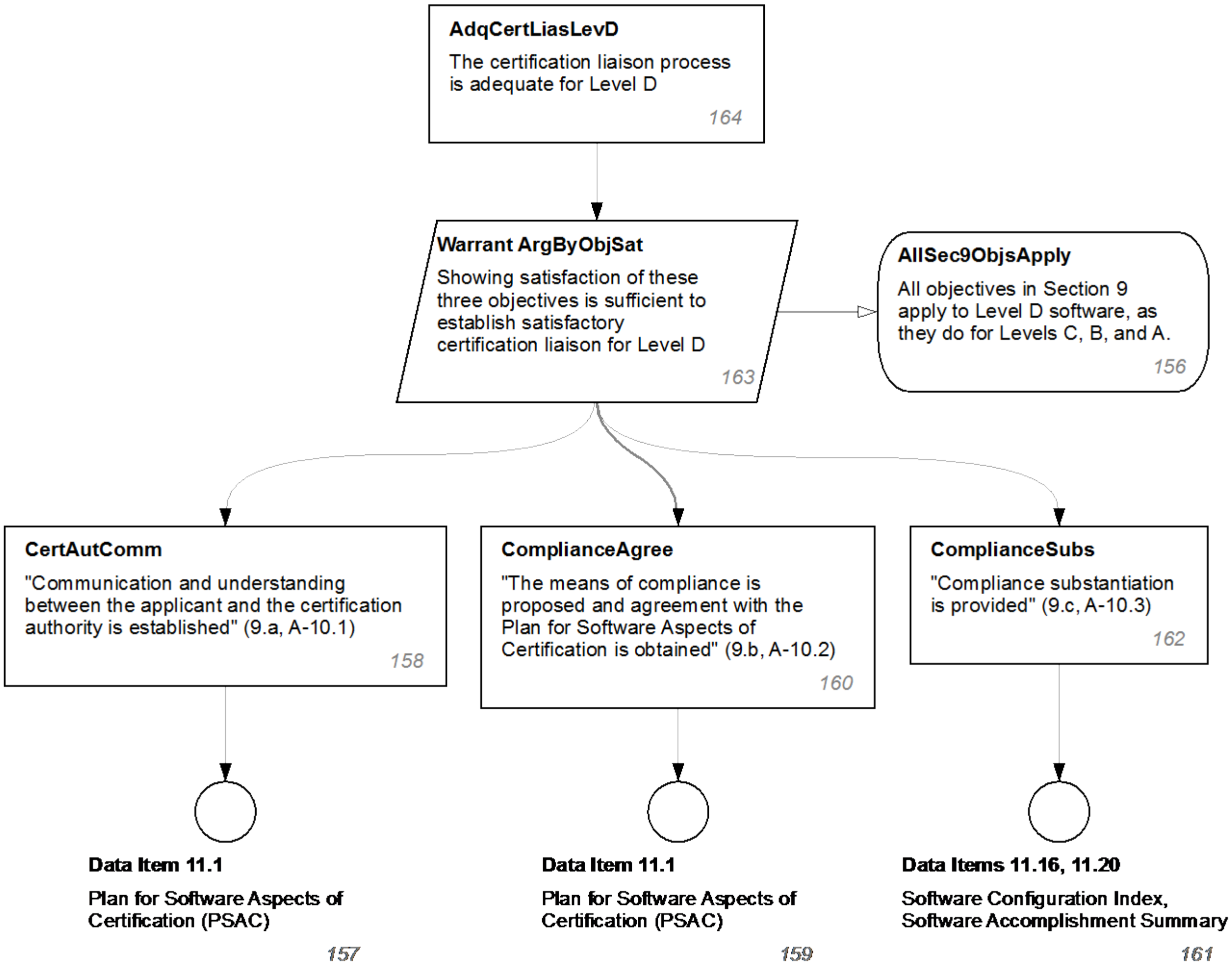

AdqCertLiasLevD
The certification liaison process is adequate for Level D
164
Warrant ArgByObjSat
Showing satisfaction of these three objectives is sufficient to establish satisfactory certification liaison for Level D
163
AllSec9ObjsApply
All objectives in Section 9 apply to Level D software, as they do for Levels C, B, and A.
156
CertAutComm
"Communication and understanding between the applicant and the certification authority is established" (9.a, A-10.1)
158
ComplianceAgree
"The means of compliance is proposed and agreement with the Plan for Software Aspects of Certification is obtained" (9.b, A-10.2)
160
ComplianceSubs
"Compliance substantiation is provided" (9.c, A-10.3)
162
Data Item 11.1
Plan for Software Aspects of Certification (PSAC)
157
Data Item 11.1
Plan for Software Aspects of Certification (PSAC)
159
Data Items 11.16, 11.20
Software Configuration Index, Software Accomplishment Summary
161

**Figure A-9. Level D: AdqCertLiasLevD**

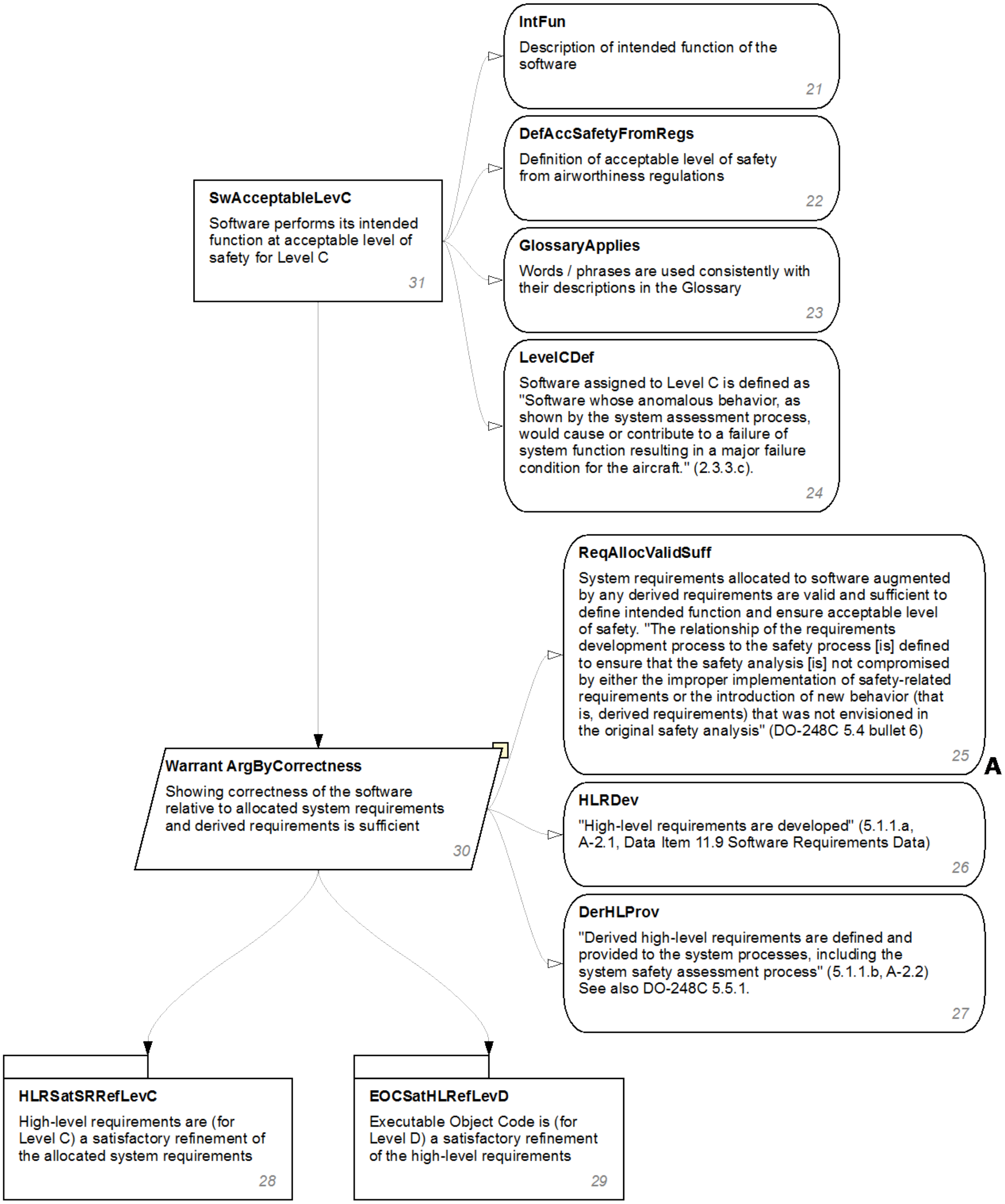

IntFun
Description of intended function of the software
21
DefAccSafetyFromRegs
Definition of acceptable level of safety from airworthiness regulations
22
SwAcceptableLevC
Software performs its intended function at acceptable level of safety for Level C
31
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
23
LevelCDef
Software assigned to Level C is defined as "Software whose anomalous behavior, as shown by the system assessment process, would cause or contribute to a failure of system function resulting in a major failure condition for the aircraft." (2.3.3.c).
24
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. "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 original safety analysis" (DO-248C 5.4 bullet 6)
25
A
Warrant ArgByCorrectness
Showing correctness of the software relative to allocated system requirements and derived requirements is sufficient
30
HLRDev
"High-level requirements are developed" (5.1.1.a, A-2.1, Data Item 11.9 Software Requirements Data)
26
DerHLProv
"Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process" (5.1.1.b, A-2.2) See also DO-248C 5.5.1.
27
HLRSatSRRefLevC
High-level requirements are (for Level C) a satisfactory refinement of the allocated system requirements
28
EOCSatHLRefLevD
Executable Object Code is (for Level D) a satisfactory refinement of the high-level requirements
29

**Figure A-10. Level C: SwAcceptableLevC**

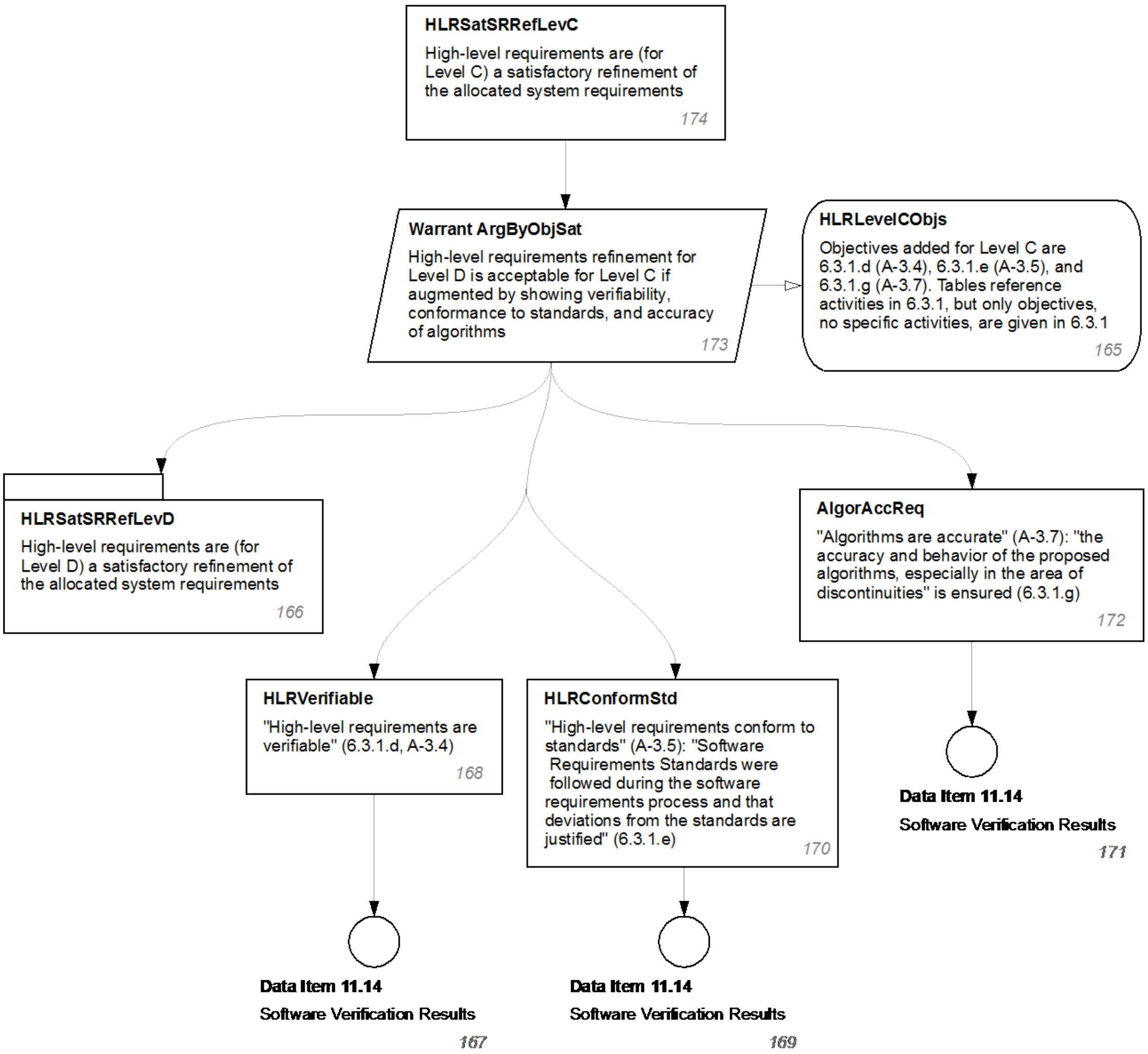

HLRSatSRRefLevC
High-level requirements are (for Level C) a satisfactory refinement of the allocated system requirements
174
Warrant ArgByObjSat
High-level requirements refinement for Level D is acceptable for Level C if augmented by showing verifiability, conformance to standards, and accuracy of algorithms
173
HLRLevelCObjs
Objectives added for Level C are 6.3.1.d (A-3.4), 6.3.1.e (A-3.5), and 6.3.1.g (A-3.7). Tables reference activities in 6.3.1, but only objectives, no specific activities, are given in 6.3.1
165
HLRSatSRRefLevD
High-level requirements are (for Level D) a satisfactory refinement of the allocated system requirements
166
AlgorAccReq
"Algorithms are accurate" (A-3.7): "the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities" is ensured (6.3.1.g)
172
HLRVerifiable
"High-level requirements are verifiable" (6.3.1.d, A-3.4)
168
HLRConformStd
"High-level requirements conform to standards" (A-3.5): "Software Requirements Standards were followed during the software requirements process and that deviations from the standards are justified" (6.3.1.e)
170
Data Item 11.14
Software Verification Results
171
Data Item 11.14
Software Verification Results
167
Data Item 11.14
Software Verification Results
169

**Figure A-11. Level C: HLRSatSRRefLevC**

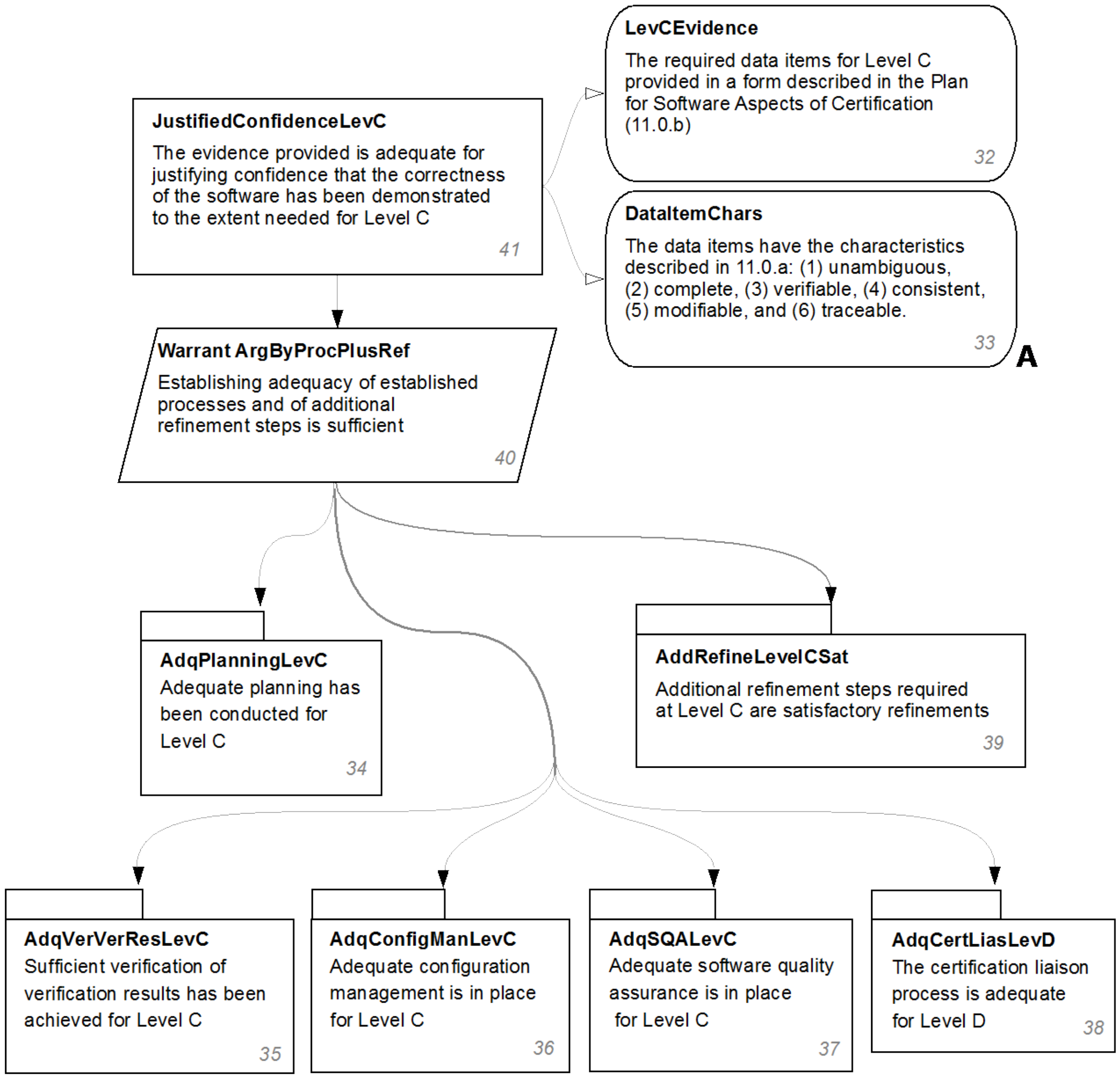

LevCEvidence
The required data items for Level C provided in a form described in the Plan for Software Aspects of Certification (11.0.b)
32
JustifiedConfidenceLevC
The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for Level C
41
DataItemChars
The data items have the characteristics described in 11.0.a: (1) unambiguous, (2) complete, (3) verifiable, (4) consistent, (5) modifiable, and (6) traceable.
33
A
Warrant ArgByProcPlusRef
Establishing adequacy of established processes and of additional refinement steps is sufficient
40
AdqPlanningLevC
Adequate planning has been conducted for Level C
34
AddRefineLevelCSat
Additional refinement steps required at Level C are satisfactory refinements
39
AdqVerVerResLevC
Sufficient verification of verification results has been achieved for Level C
35
AdqConfigManLevC
Adequate configuration management is in place for Level C
36
AdqSQALevC
Adequate software quality assurance is in place for Level C
37
AdqCertLiasLevD
The certification liaison process is adequate for Level D
38

**Figure A-12. Level C: JustifiedConfidenceLevC**

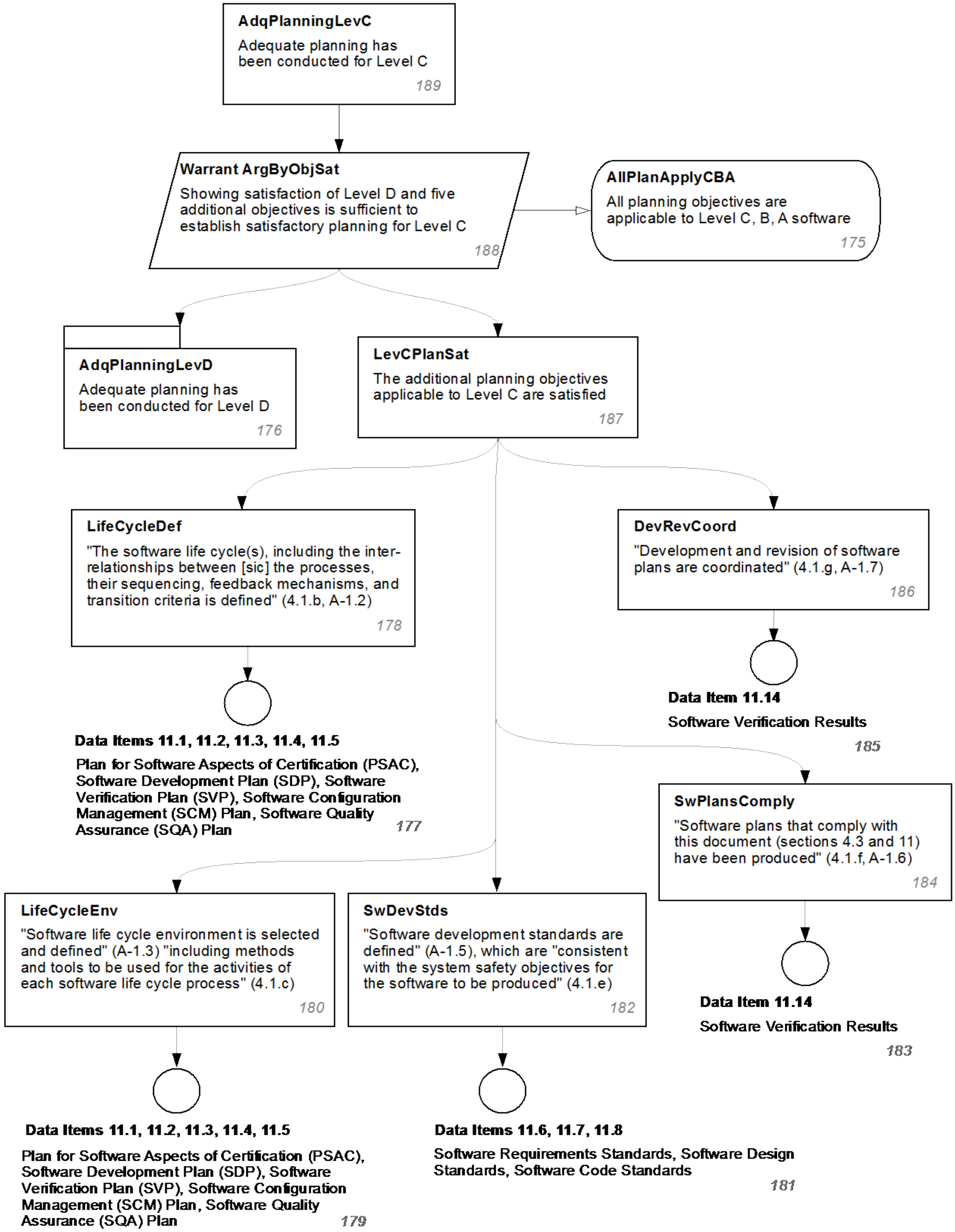

AdqPlanningLevC
Adequate planning has been conducted for Level C
189
Warrant ArgByObjSat
Showing satisfaction of Level D and five additional objectives is sufficient to establish satisfactory planning for Level C
188
AllPlanApplyCBA
All planning objectives are applicable to Level C, B, A software
175
AdqPlanningLevD
Adequate planning has been conducted for Level D
176
LevCPlanSat
The additional planning objectives applicable to Level C are satisfied
187
LifeCycleDef
"The software life cycle(s), including the inter-relationships between [sic] the processes, their sequencing, feedback mechanisms, and transition criteria is defined" (4.1.b, A-1.2)
178
DevRevCoord
"Development and revision of software plans are coordinated" (4.1.g, A-1.7)
186
Data Item 11.14
Software Verification Results
185
Data Items 11.1, 11.2, 11.3, 11.4, 11.5
Plan for Software Aspects of Certification (PSAC), Software Development Plan (SDP), Software Verification Plan (SVP), Software Configuration Management (SCM) Plan, Software Quality Assurance (SQA) Plan
177
SwPlansComply
"Software plans that comply with this document (sections 4.3 and 11) have been produced" (4.1.f, A-1.6)
184
LifeCycleEnv
"Software life cycle environment is selected and defined" (A-1.3) "including methods and tools to be used for the activities of each software life cycle process" (4.1.c)
180
SwDevStds
"Software development standards are defined" (A-1.5), which are "consistent with the system safety objectives for the software to be produced" (4.1.e)
182
Data Item 11.14
Software Verification Results
183
Data Items 11.1, 11.2, 11.3, 11.4, 11.5
Plan for Software Aspects of Certification (PSAC), Software Development Plan (SDP), Software Verification Plan (SVP), Software Configuration Management (SCM) Plan, Software Quality Assurance (SQA) Plan
179
Data Items 11.6, 11.7, 11.8
Software Requirements Standards, Software Design Standards, Software Code Standards
181

**Figure A-13. Level C: AdqPlanningLevC**

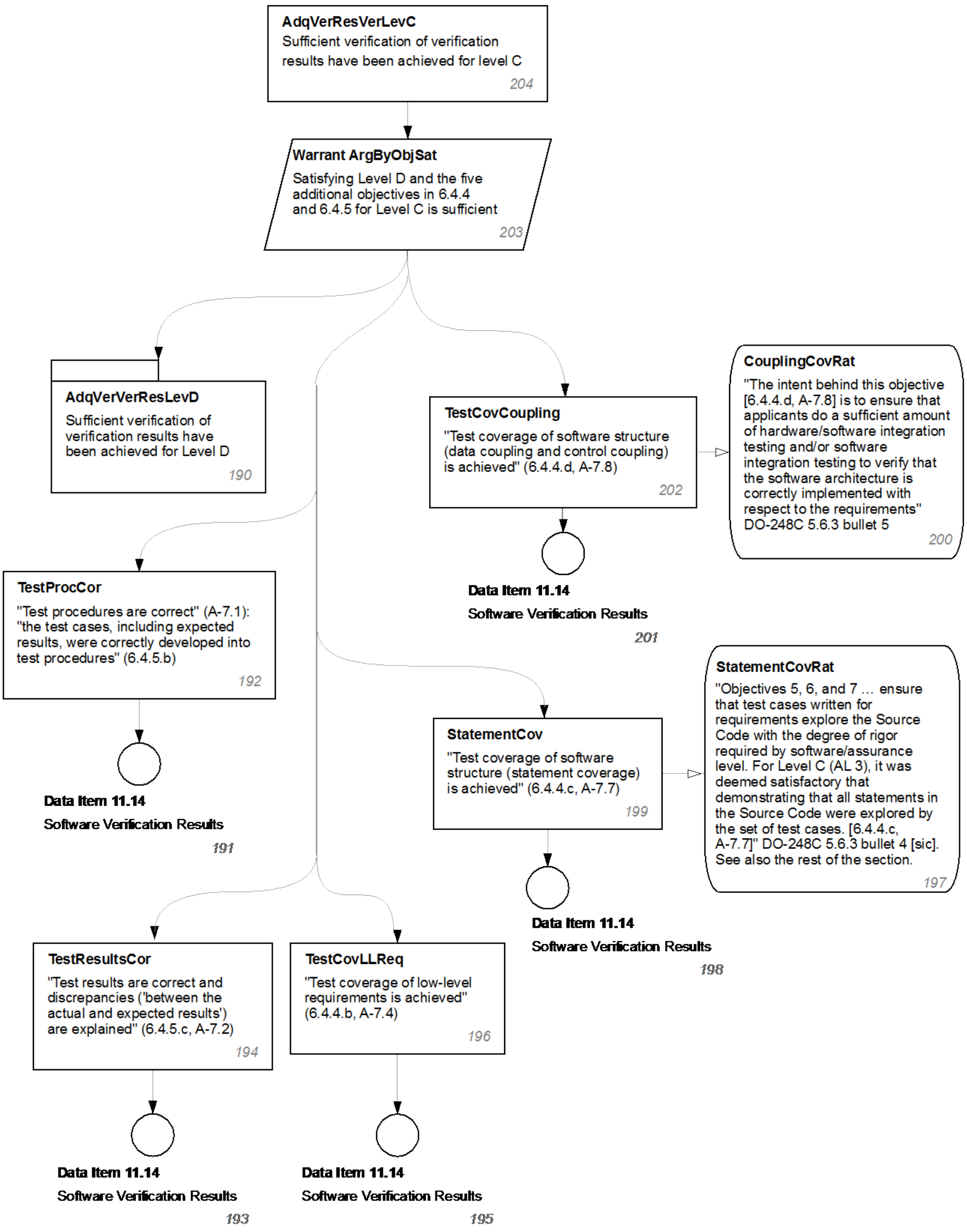

AdqVerResVerLevC
Sufficient verification of verification results have been achieved for level C
204
Warrant ArgByObjSat
Satisfying Level D and the five additional objectives in 6.4.4 and 6.4.5 for Level C is sufficient
203
AdqVerVerResLevD
Sufficient verification of verification results have been achieved for Level D
190
TestCovCoupling
"Test coverage of software structure (data coupling and control coupling) is achieved" (6.4.4.d, A-7.8)
202
CouplingCovRat
"The intent behind this objective [6.4.4.d, A-7.8] is to ensure that applicants do a sufficient amount of hardware/software integration testing and/or software integration testing to verify that the software architecture is correctly implemented with respect to the requirements" DO-248C 5.6.3 bullet 5
200
Data Item 11.14
Software Verification Results
201
TestProcCor
"Test procedures are correct" (A-7.1): "the test cases, including expected results, were correctly developed into test procedures" (6.4.5.b)
192
StatementCovRat
"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 C (AL 3), it was deemed satisfactory that demonstrating that all statements in the Source Code were explored by the set of test cases. [6.4.4.c, A-7.7]" DO-248C 5.6.3 bullet 4 [sic]. See also the rest of the section.
197
StatementCov
"Test coverage of software structure (statement coverage) is achieved" (6.4.4.c, A-7.7)
199
Data Item 11.14
Software Verification Results
191
Data Item 11.14
Software Verification Results
198
TestResultsCor
"Test results are correct and discrepancies ('between the actual and expected results') are explained" (6.4.5.c, A-7.2)
194
TestCovLLReq
"Test coverage of low-level requirements is achieved" (6.4.4.b, A-7.4)
196
Data Item 11.14
Software Verification Results
193
Data Item 11.14
Software Verification Results
195

**Figure A-14. Level C: AdqVerVerLevC**

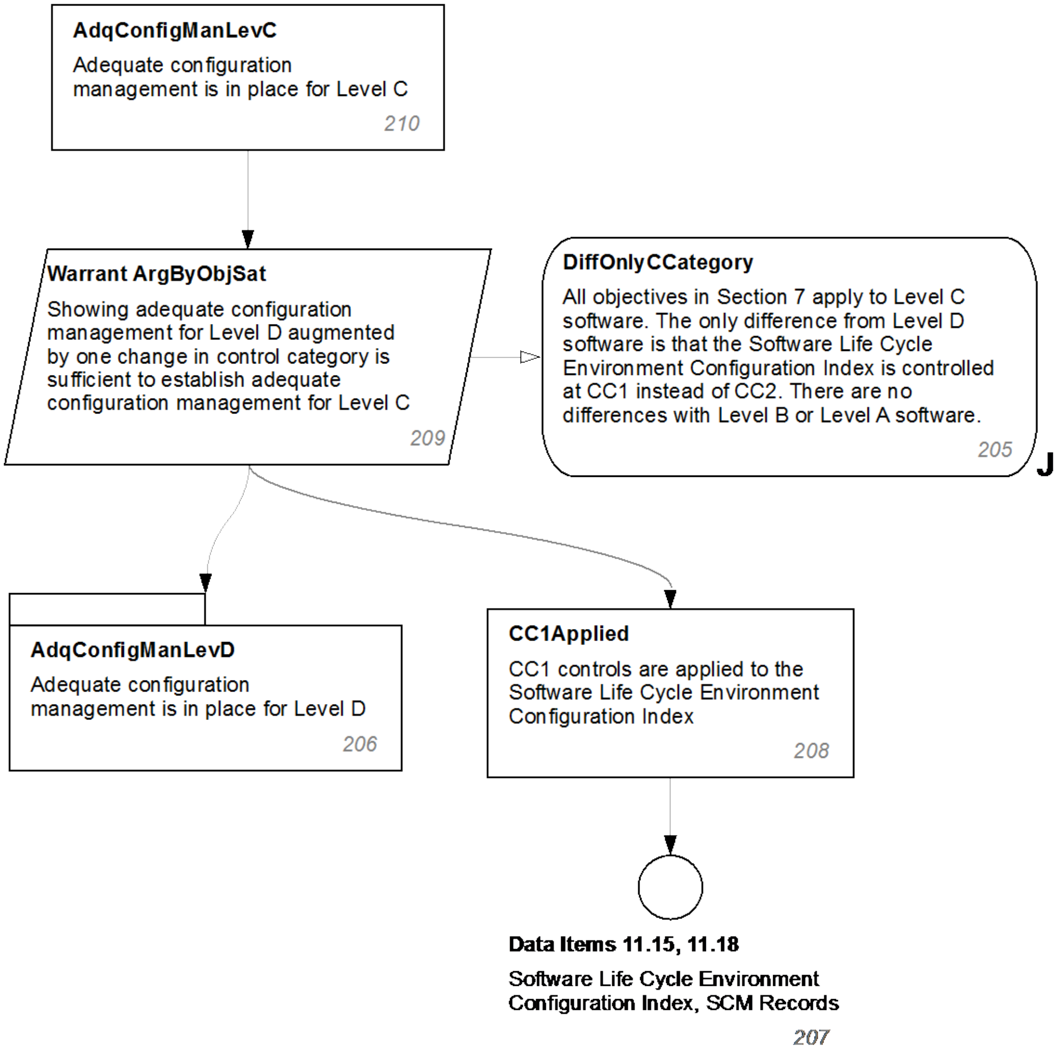

AdqConfigManLevC
Adequate configuration management is in place for Level C
210
Warrant ArgByObjSat
Showing adequate configuration management for Level D augmented by one change in control category is sufficient to establish adequate configuration management for Level C
209
DiffOnlyCCategory
All objectives in Section 7 apply to Level C software. The only difference from Level D software is that the Software Life Cycle Environment Configuration Index is controlled at CC1 instead of CC2. There are no differences with Level B or Level A software.
205
J
AdqConfigManLevD
Adequate configuration management is in place for Level D
206
CC1Applied
CC1 controls are applied to the Software Life Cycle Environment Configuration Index
208
Data Items 11.15, 11.18
Software Life Cycle Environment Configuration Index, SCM Records
207

**Figure A-15. Level C: AdqConfigManLevC**

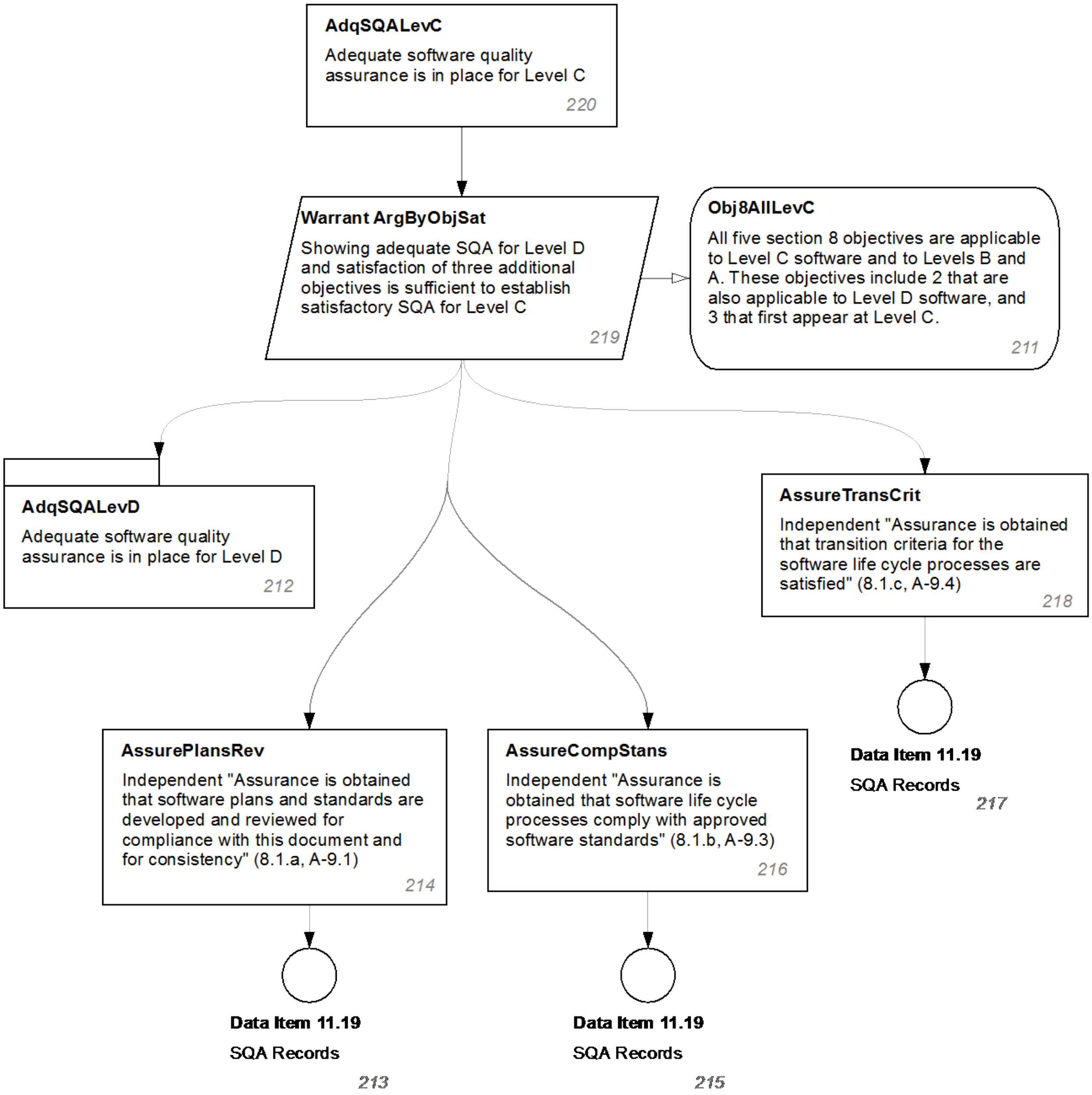

AdqSQALevC
Adequate software quality assurance is in place for Level C
220
Warrant ArgByObjSat
Showing adequate SQA for Level D and satisfaction of three additional objectives is sufficient to establish satisfactory SQA for Level C
219
Obj8AllLevC
All five section 8 objectives are applicable to Level C software and to Levels B and A. These objectives include 2 that are also applicable to Level D software, and 3 that first appear at Level C.
211
AdqSQALevD
Adequate software quality assurance is in place for Level D
212
AssureTransCrit
Independent "Assurance is obtained that transition criteria for the software life cycle processes are satisfied" (8.1.c, A-9.4)
218
Data Item 11.19
SQA Records
217
AssurePlansRev
Independent "Assurance is obtained that software plans and standards are developed and reviewed for compliance with this document and for consistency" (8.1.a, A-9.1)
214
AssureCompStans
Independent "Assurance is obtained that software life cycle processes comply with approved software standards" (8.1.b, A-9.3)
216
Data Item 11.19
SQA Records
213
Data Item 11.19
SQA Records
215

Figure A-16. Level C: AdqSQALevC

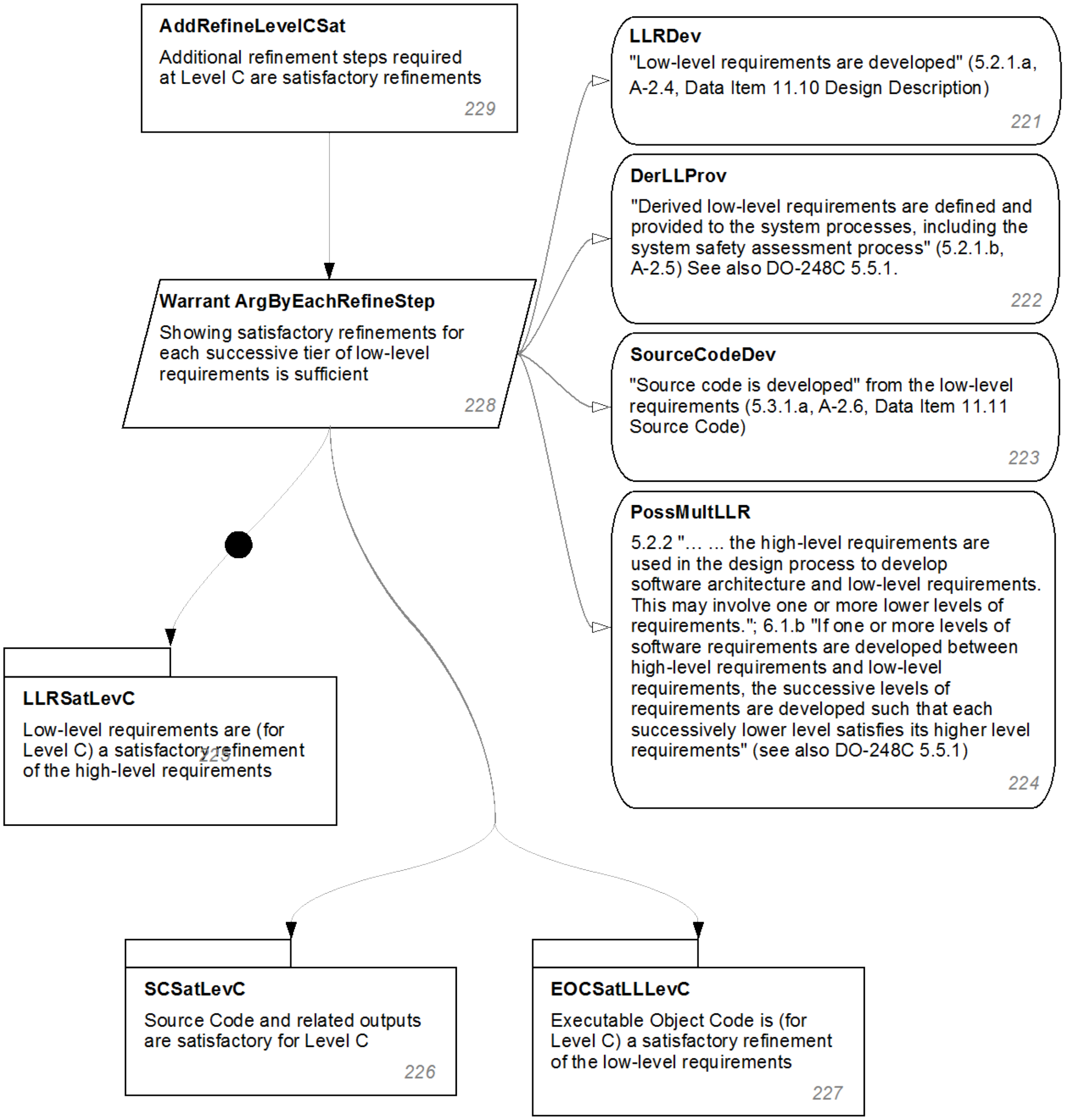

AddRefineLevelCSat
Additional refinement steps required at Level C are satisfactory refinements
229
Warrant ArgByEachRefineStep
Showing satisfactory refinements for each successive tier of low-level requirements is sufficient
228
LLRDev
"Low-level requirements are developed" (5.2.1.a, A-2.4, Data Item 11.10 Design Description)
221
DerLLProv
"Derived low-level requirements are defined and provided to the system processes, including the system safety assessment process" (5.2.1.b, A-2.5) See also DO-248C 5.5.1.
222
SourceCodeDev
"Source code is developed" from the low-level requirements (5.3.1.a, A-2.6, Data Item 11.11 Source Code)
223
PossMultLLR
5.2.2 "… … the high-level requirements are used in the design process to develop software architecture and low-level requirements. This may involve one or more lower levels of requirements."; 6.1.b "If one or more levels of software requirements are developed between high-level requirements and low-level requirements, the successive levels of requirements are developed such that each successively lower level satisfies its higher level requirements" (see also DO-248C 5.5.1)
224
LLRSatLevC
Low-level requirements are (for Level C) a satisfactory refinement of the high-level requirements
225
SCSatLevC
Source Code and related outputs are satisfactory for Level C
226
EOCSatLLLevC
Executable Object Code is (for Level C) a satisfactory refinement of the low-level requirements
227

**Figure A-17. Level C: AddRefineLevelCSat**

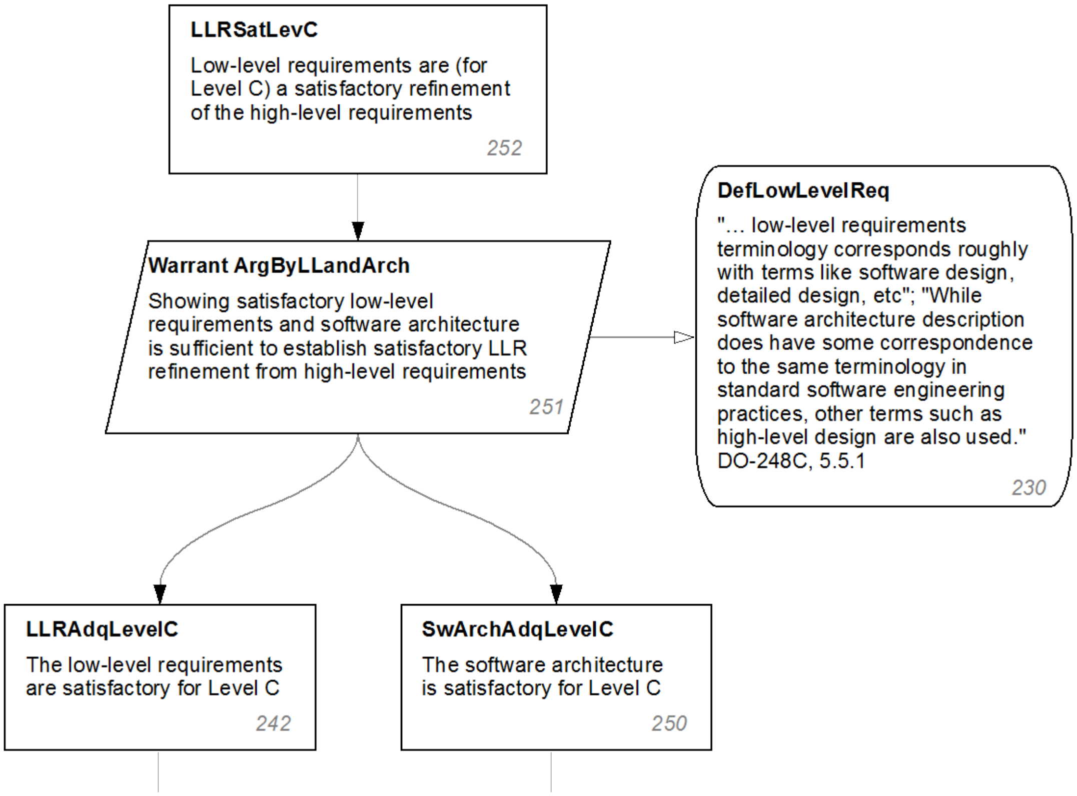

LLRSatLevC
Low-level requirements are (for Level C) a satisfactory refinement of the high-level requirements
252
Warrant ArgByLLandArch
Showing satisfactory low-level requirements and software architecture is sufficient to establish satisfactory LLR refinement from high-level requirements
251
DefLowLevelReq
"... low-level requirements terminology corresponds roughly with terms like software design, detailed design, etc"; "While software architecture description does have some correspondence to the same terminology in standard software engineering practices, other terms such as high-level design are also used." DO-248C, 5.5.1
230
LLRAdqLevelC
The low-level requirements are satisfactory for Level C
242
SwArchAdqLevelC
The software architecture is satisfactory for Level C
250

**Figure A-18. Level C: LLRSatLevC**

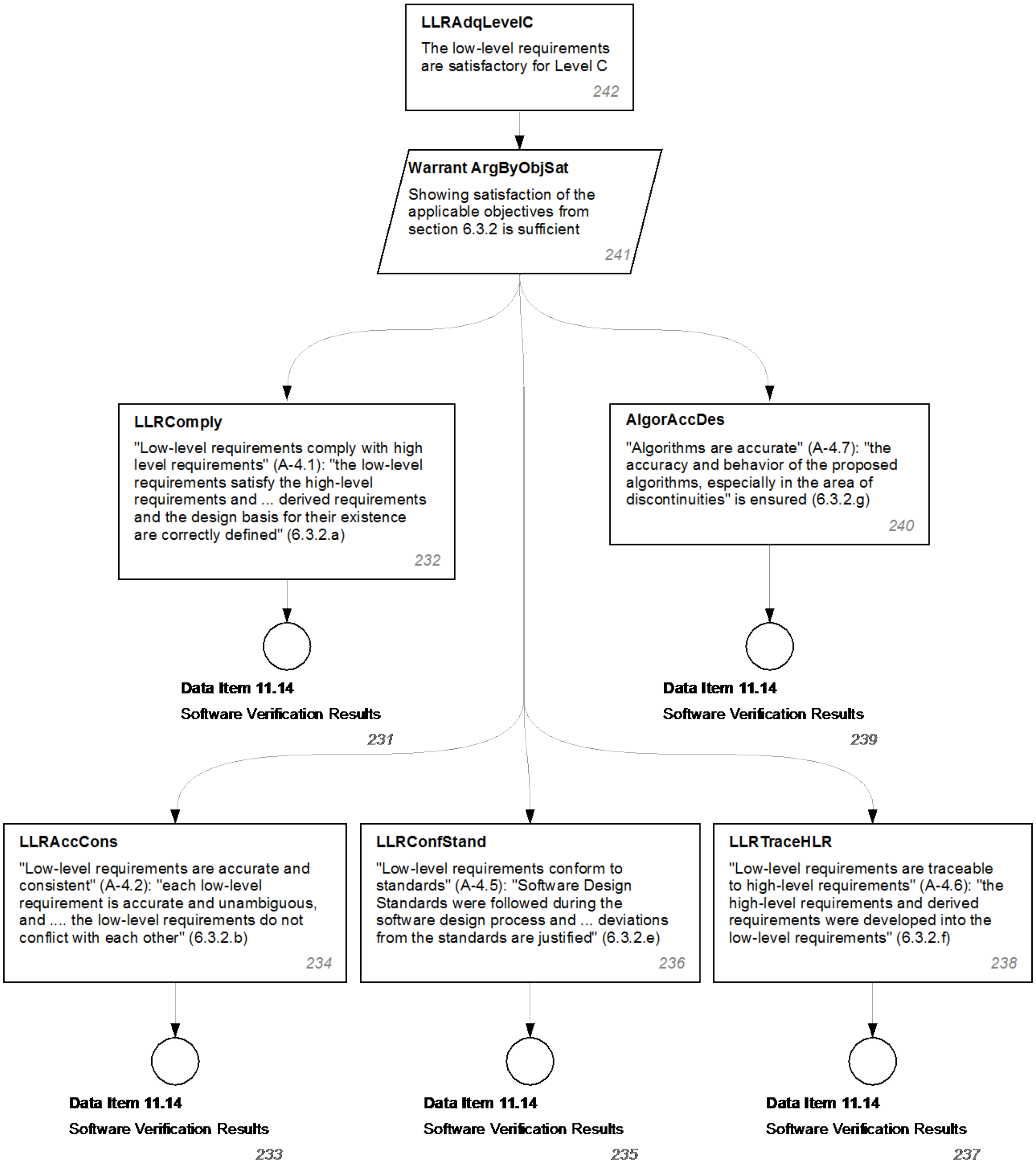

LLRAdqLevelC
The low-level requirements are satisfactory for Level C
242
Warrant ArgByObjSat
Showing satisfaction of the applicable objectives from section 6.3.2 is sufficient
241
LLRComply
"Low-level requirements comply with high level requirements" (A-4.1): "the low-level requirements satisfy the high-level requirements and ... derived requirements and the design basis for their existence are correctly defined" (6.3.2.a)
232
AlgorAccDes
"Algorithms are accurate" (A-4.7): "the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities" is ensured (6.3.2.g)
240
Data Item 11.14
Software Verification Results
231
Data Item 11.14
Software Verification Results
239
LLRAccCons
"Low-level requirements are accurate and consistent" (A-4.2): "each low-level requirement is accurate and unambiguous, and .... the low-level requirements do not conflict with each other" (6.3.2.b)
234
LLRConfStand
"Low-level requirements conform to standards" (A-4.5): "Software Design Standards were followed during the software design process and ... deviations from the standards are justified" (6.3.2.e)
236
LLRTraceHLR
"Low-level requirements are traceable to high-level requirements" (A-4.6): "the high-level requirements and derived requirements were developed into the low-level requirements" (6.3.2.f)
238
Data Item 11.14
Software Verification Results
233
Data Item 11.14
Software Verification Results
235
Data Item 11.14
Software Verification Results
237

**Figure A-19. Level C: LLRSatLevC/LLRAdqLevelC**

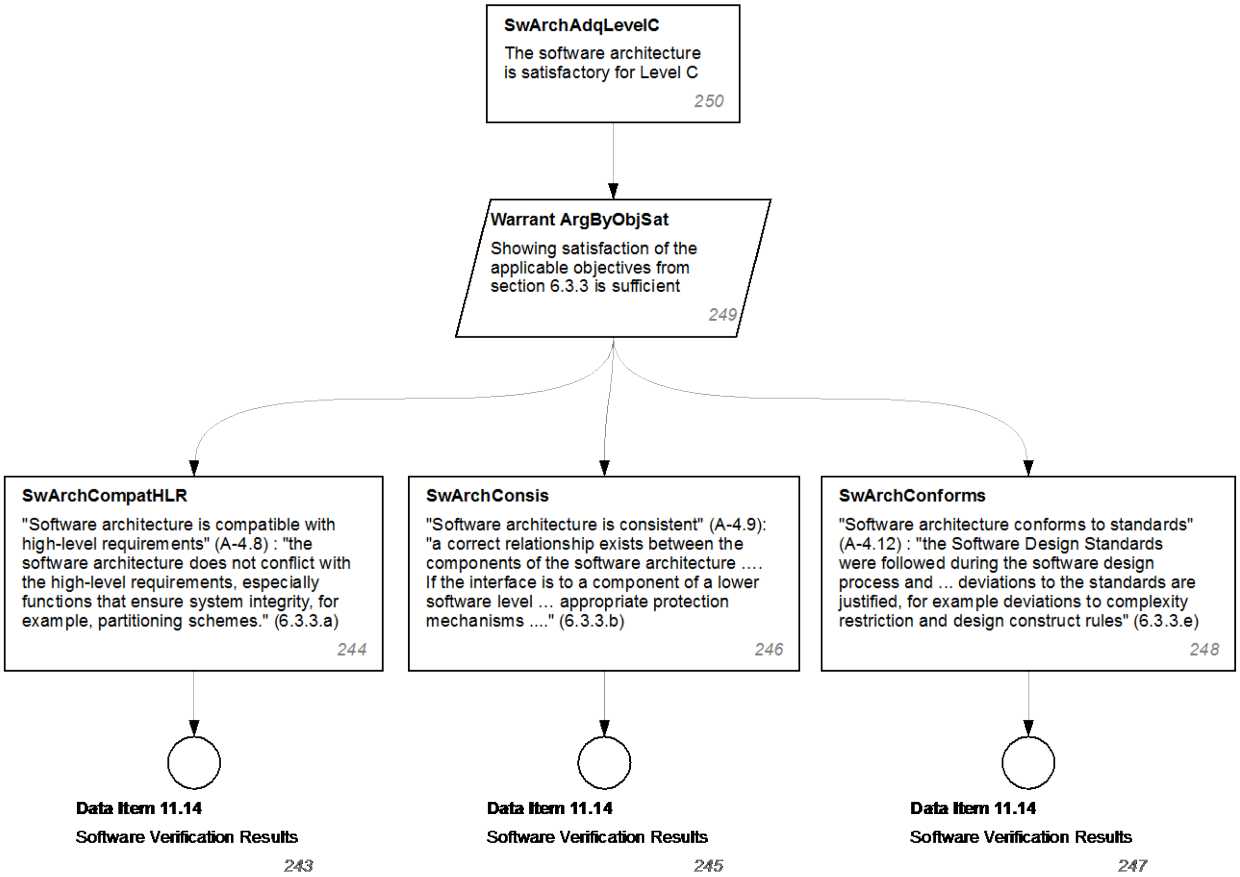

SwArchAdqLevelC
The software architecture is satisfactory for Level C
250
Warrant ArgByObjSat
Showing satisfaction of the applicable objectives from section 6.3.3 is sufficient
249
SwArchCompatHLR
"Software architecture is compatible with high-level requirements" (A-4.8) : "the software architecture does not conflict with the high-level requirements, especially functions that ensure system integrity, for example, partitioning schemes." (6.3.3.a)
244
SwArchConsis
"Software architecture is consistent" (A-4.9): "a correct relationship exists between the components of the software architecture …. If the interface is to a component of a lower software level … appropriate protection mechanisms …." (6.3.3.b)
246
SwArchConforms
"Software architecture conforms to standards" (A-4.12) : "the Software Design Standards were followed during the software design process and … deviations to the standards are justified, for example deviations to complexity restriction and design construct rules" (6.3.3.e)
248
Data Item 11.14
Software Verification Results
243
Data Item 11.14
Software Verification Results
245
Data Item 11.14
Software Verification Results
247

**Figure A-20. Level C: LLRSatLevC/SwArchAdqLevelC**

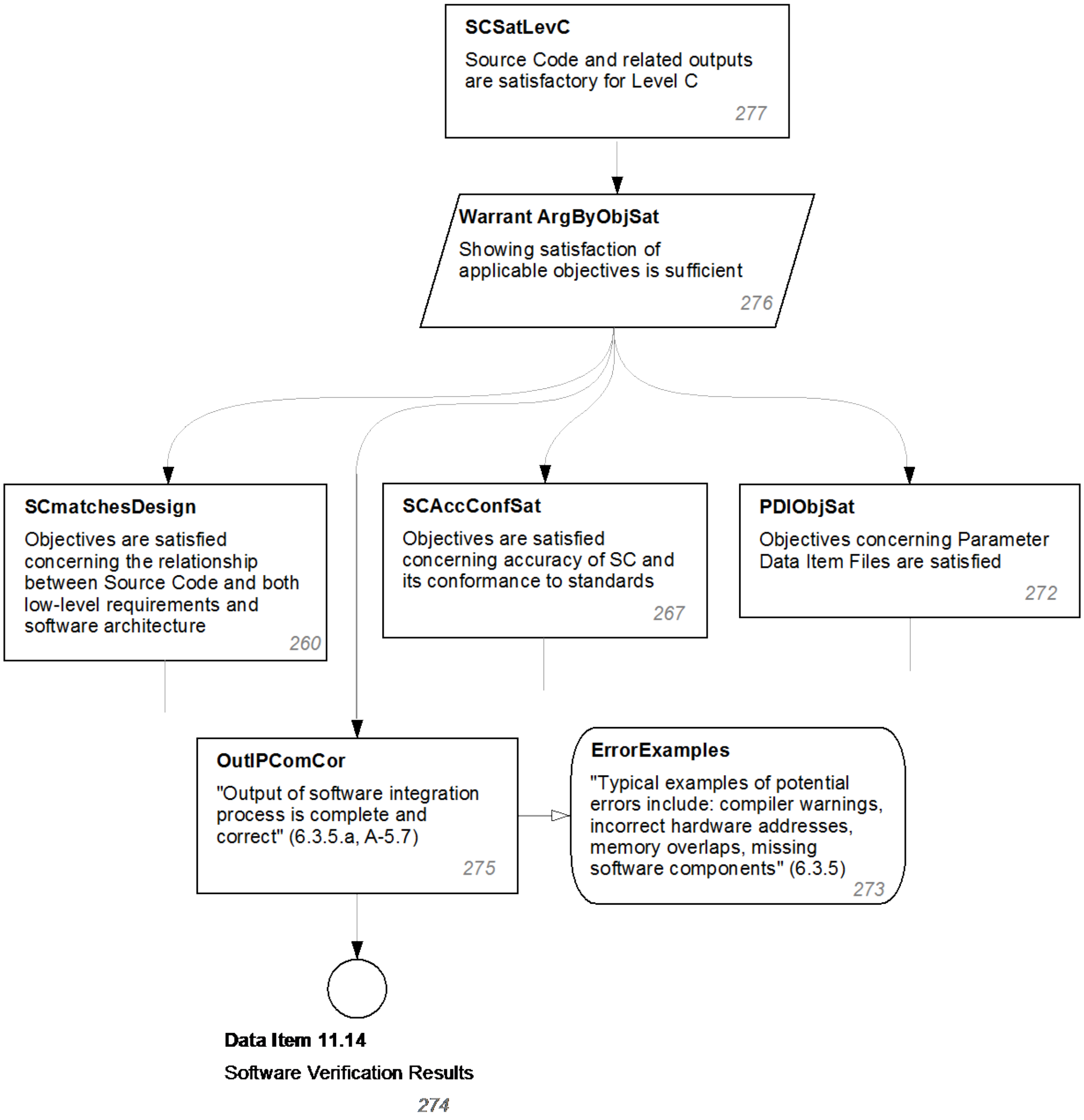

SCSatLevC
Source Code and related outputs are satisfactory for Level C
277
Warrant ArgByObjSat
Showing satisfaction of applicable objectives is sufficient
276
SCmatchesDesign
Objectives are satisfied concerning the relationship between Source Code and both low-level requirements and software architecture
260
SCAccConfSat
Objectives are satisfied concerning accuracy of SC and its conformance to standards
267
PDIObjSat
Objectives concerning Parameter Data Item Files are satisfied
272
OutIPComCor
"Output of software integration process is complete and correct" (6.3.5.a, A-5.7)
275
ErrorExamples
"Typical examples of potential errors include: compiler warnings, incorrect hardware addresses, memory overlaps, missing software components" (6.3.5)
273
Data Item 11.14
Software Verification Results
274

**Figure A-21. Level C: SCSatLevC**

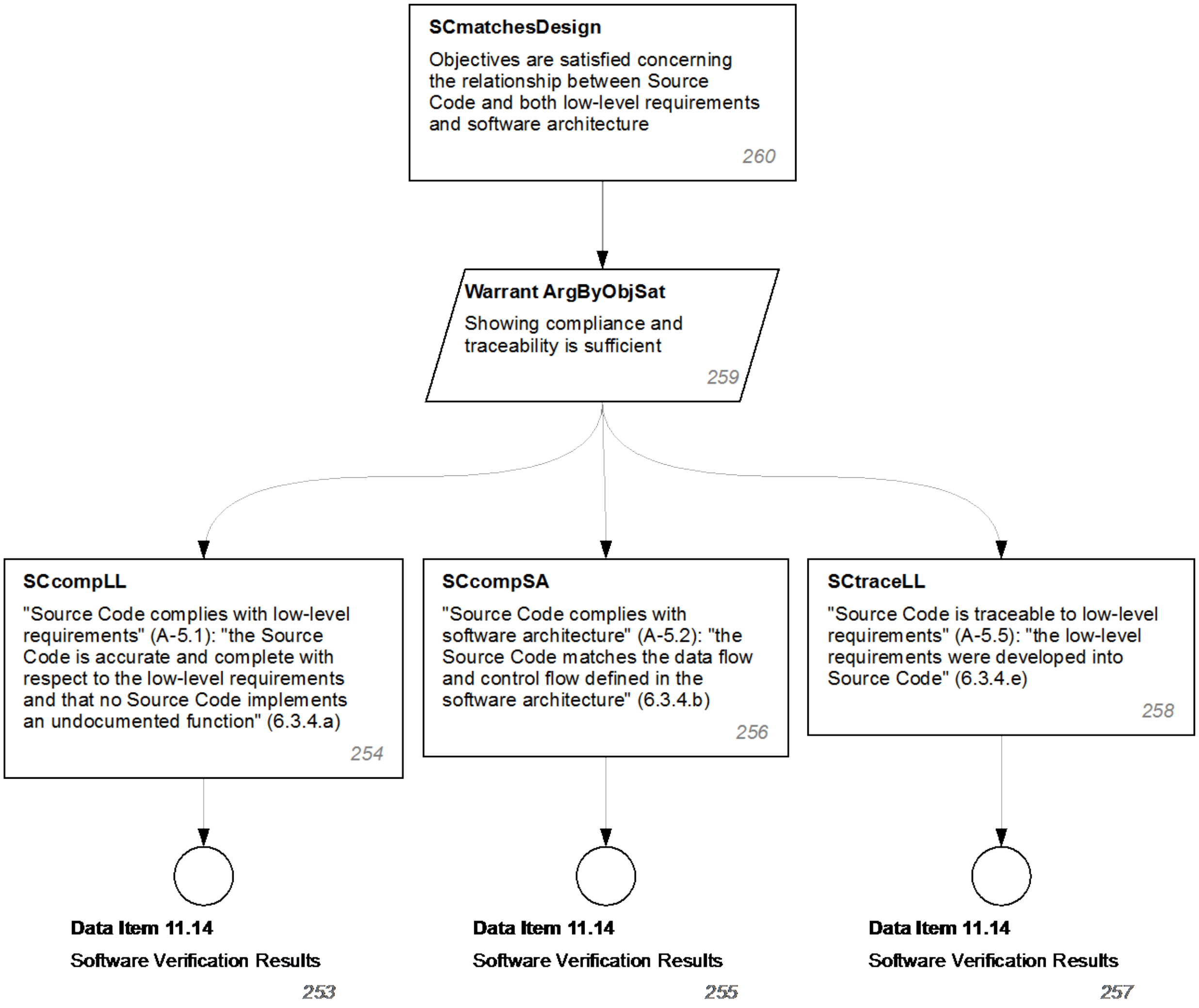

SCmatchesDesign
Objectives are satisfied concerning the relationship between Source Code and both low-level requirements and software architecture
260
Warrant ArgByObjSat
Showing compliance and traceability is sufficient
259
SCcompLL
"Source Code complies with low-level requirements" (A-5.1): "the Source Code is accurate and complete with respect to the low-level requirements and that no Source Code implements an undocumented function" (6.3.4.a)
254
SCcompSA
"Source Code complies with software architecture" (A-5.2): "the Source Code matches the data flow and control flow defined in the software architecture" (6.3.4.b)
256
SCtraceLL
"Source Code is traceable to low-level requirements" (A-5.5): "the low-level requirements were developed into Source Code" (6.3.4.e)
258
Data Item 11.14
Software Verification Results
253
Data Item 11.14
Software Verification Results
255
Data Item 11.14
Software Verification Results
257

**Figure A-22. Level C: SCSatLevC/ScmatchesDesign**

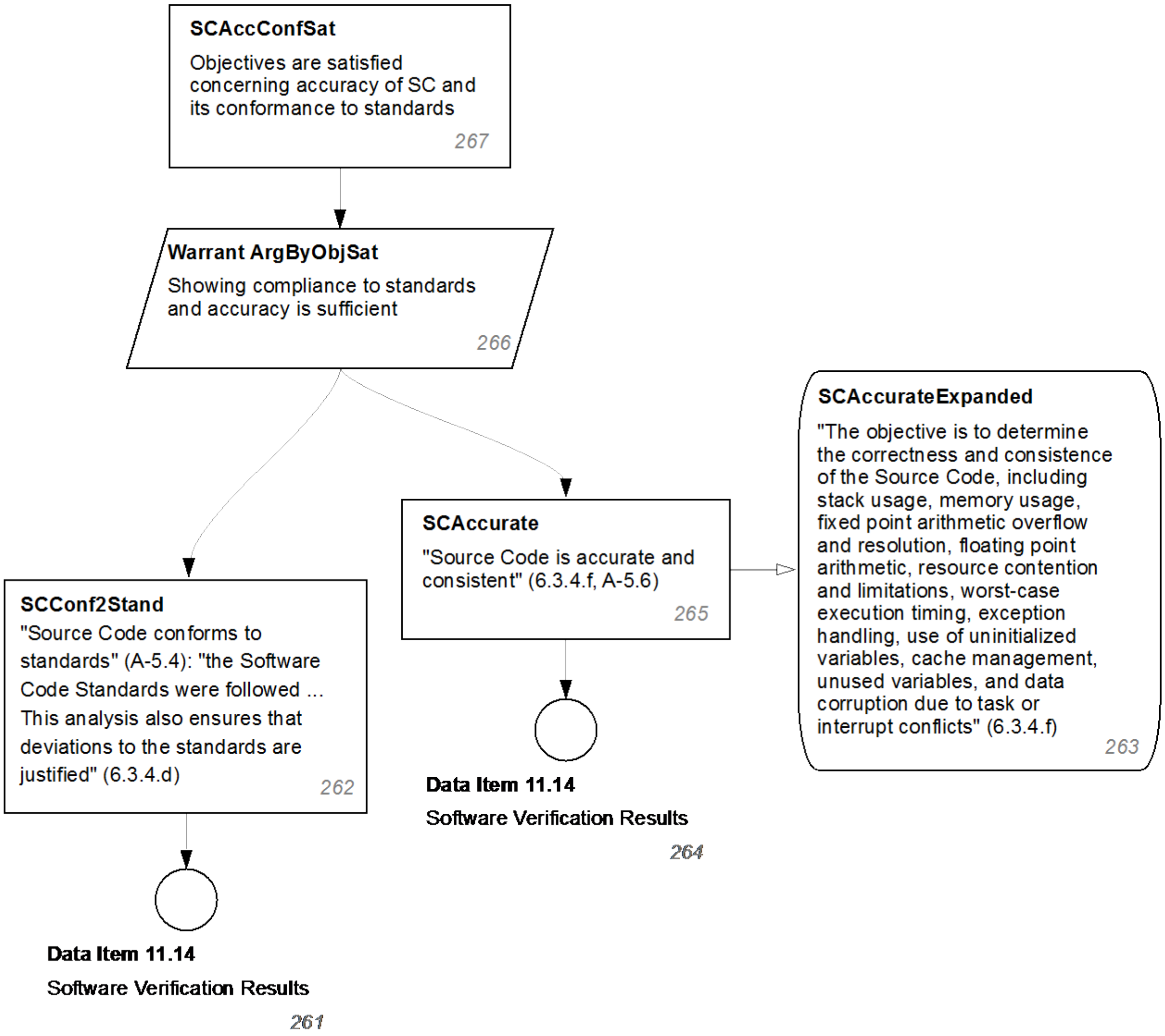

SCAccConfSat
Objectives are satisfied concerning accuracy of SC and its conformance to standards
267
Warrant ArgByObjSat
Showing compliance to standards and accuracy is sufficient
266
SCAccurateExpanded
"The objective is to determine the correctness and consistence of the Source Code, including stack usage, memory usage, fixed point arithmetic overflow and resolution, floating point arithmetic, resource contention and limitations, worst-case execution timing, exception handling, use of uninitialized variables, cache management, unused variables, and data corruption due to task or interrupt conflicts" (6.3.4.f)
263
SCAccurate
"Source Code is accurate and consistent" (6.3.4.f, A-5.6)
265
SCConf2Stand
"Source Code conforms to standards" (A-5.4): "the Software Code Standards were followed ... This analysis also ensures that deviations to the standards are justified" (6.3.4.d)
262
Data Item 11.14
Software Verification Results
264
Data Item 11.14
Software Verification Results
261

**Figure A-23. Level C: SCSatLevC/SCAccConfSat**

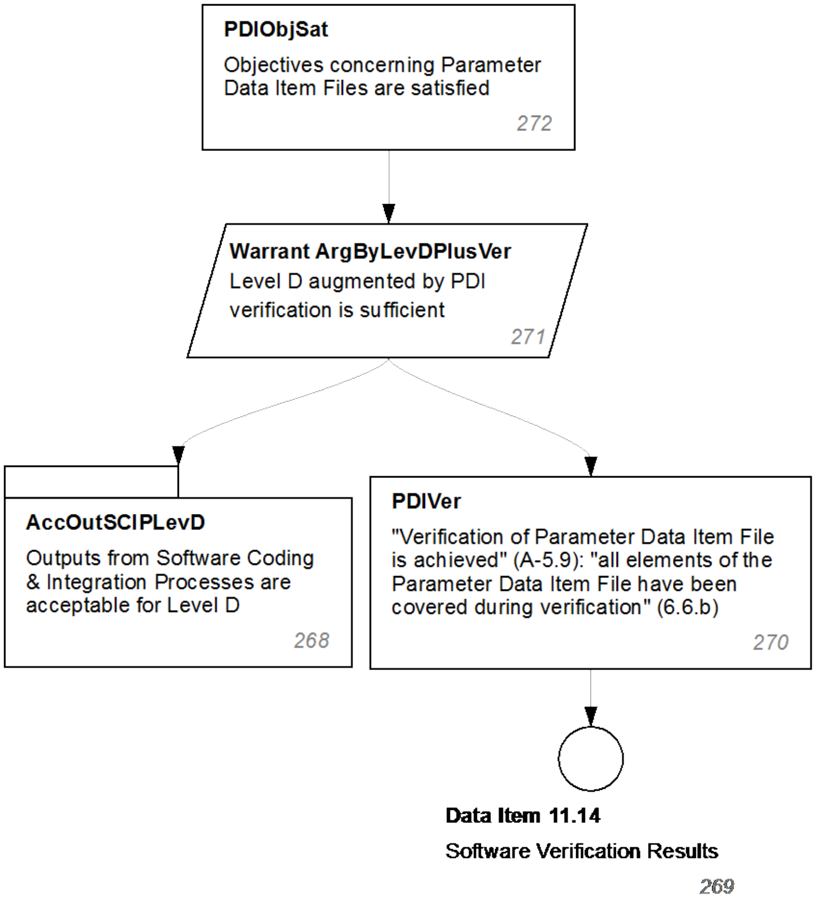

PDIObjSat
Objectives concerning Parameter Data Item Files are satisfied
272
Warrant ArgByLevDPlusVer
Level D augmented by PDI verification is sufficient
271
AccOutSCIPLevD
Outputs from Software Coding & Integration Processes are acceptable for Level D
268
PDIVer
"Verification of Parameter Data Item File is achieved" (A-5.9): "all elements of the Parameter Data Item File have been covered during verification" (6.6.b)
270
Data Item 11.14
Software Verification Results
269

**Figure A-24. Level C: SCSatLevC/PDIObjSat**

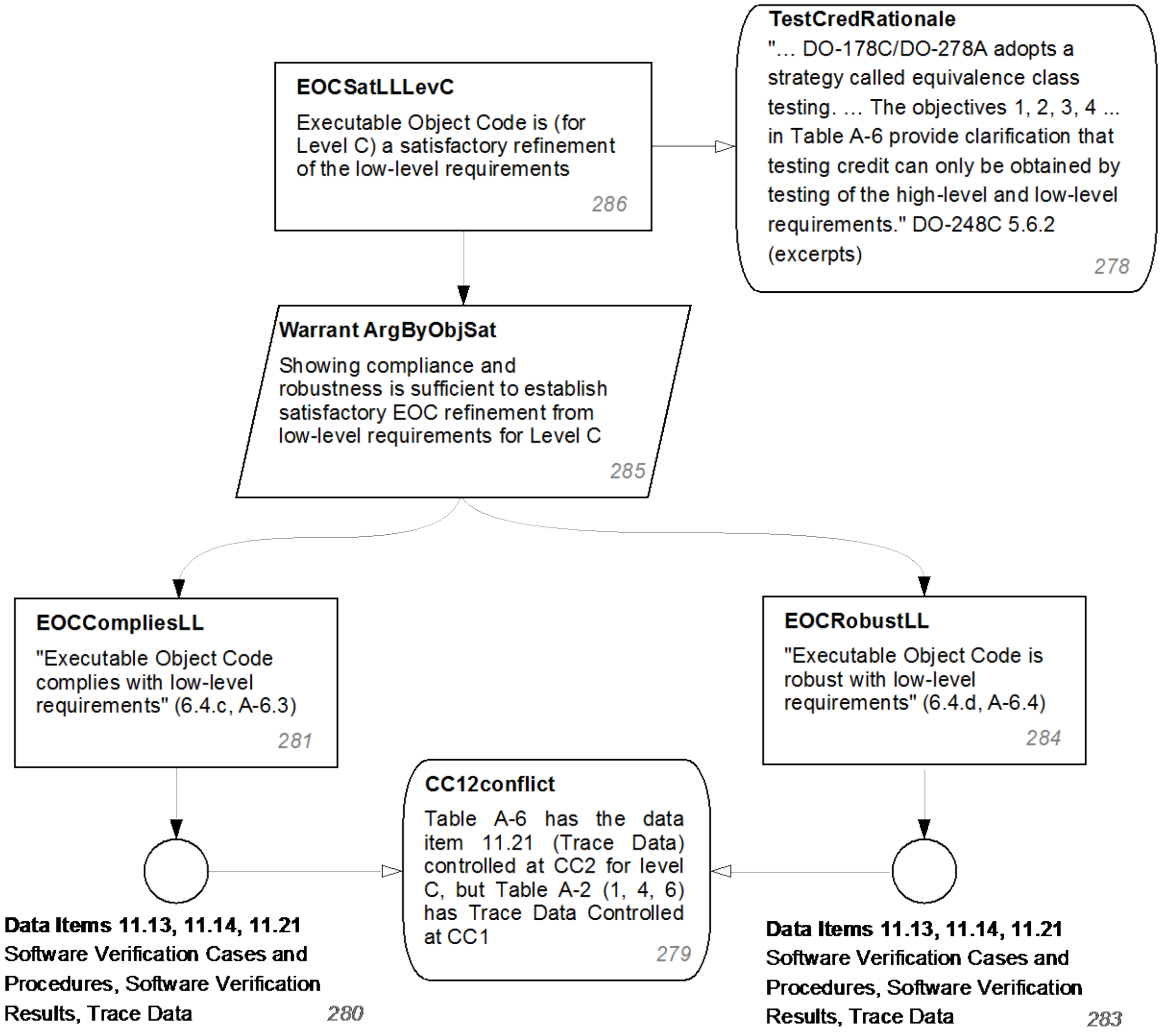

TestCredRationale
"... DO-178C/DO-278A adopts a strategy called equivalence class testing. ... The objectives 1, 2, 3, 4 ... in Table A-6 provide clarification that testing credit can only be obtained by testing of the high-level and low-level requirements." DO-248C 5.6.2 (excerpts)
278
EOCSatLLLevC
Executable Object Code is (for Level C) a satisfactory refinement of the low-level requirements
286
Warrant ArgByObjSat
Showing compliance and robustness is sufficient to establish satisfactory EOC refinement from low-level requirements for Level C
285
EOCCompliesLL
"Executable Object Code complies with low-level requirements" (6.4.c, A-6.3)
281
EOCRobustLL
"Executable Object Code is robust with low-level requirements" (6.4.d, A-6.4)
284
CC12conflict
Table A-6 has the data item 11.21 (Trace Data) controlled at CC2 for level C, but Table A-2 (1, 4, 6) has Trace Data Controlled at CC1
279
Data Items 11.13, 11.14, 11.21
Software Verification Cases and Procedures, Software Verification Results, Trace Data
280
Data Items 11.13, 11.14, 11.21
Software Verification Cases and Procedures, Software Verification Results, Trace Data
283

**Figure A-25. Level C: EOCSatLLLevC**

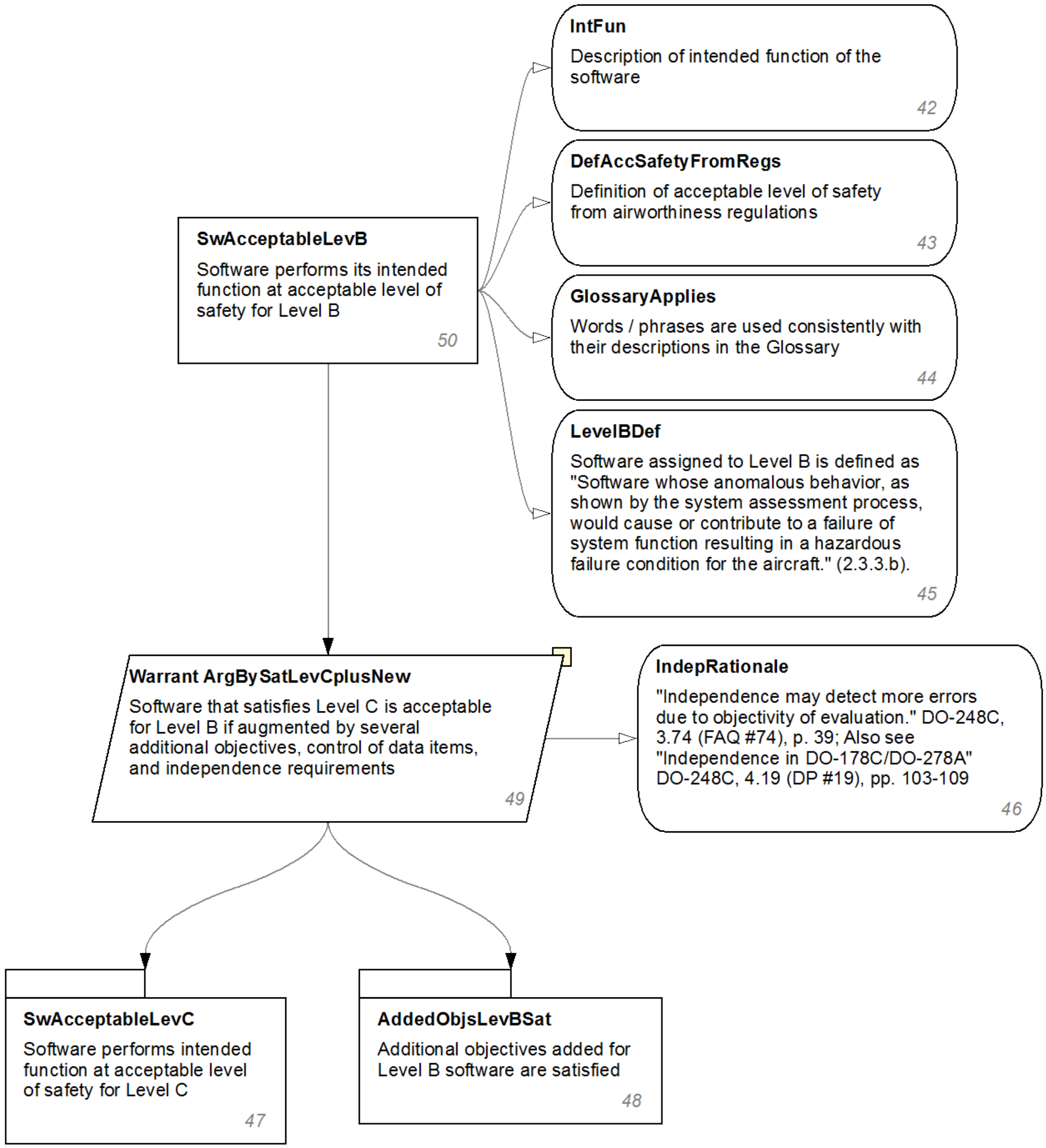

IntFun
Description of intended function of the software
42
DefAccSafetyFromRegs
Definition of acceptable level of safety from airworthiness regulations
43
SwAcceptableLevB
Software performs its intended function at acceptable level of safety for Level B
50
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
44
LevelBDef
Software assigned to Level B is defined as "Software whose anomalous behavior, as shown by the system assessment process, would cause or contribute to a failure of system function resulting in a hazardous failure condition for the aircraft." (2.3.3.b).
45
Warrant ArgBySatLevCplusNew
Software that satisfies Level C is acceptable for Level B if augmented by several additional objectives, control of data items, and independence requirements
49
IndepRationale
"Independence may detect more errors due to objectivity of evaluation." DO-248C, 3.74 (FAQ #74), p. 39; Also see "Independence in DO-178C/DO-278A" DO-248C, 4.19 (DP #19), pp. 103-109
46
SwAcceptableLevC
Software performs intended function at acceptable level of safety for Level C
47
AddedObjsLevBSat
Additional objectives added for Level B software are satisfied
48

**Figure A-26. Level B: SwAcceptableLevB**

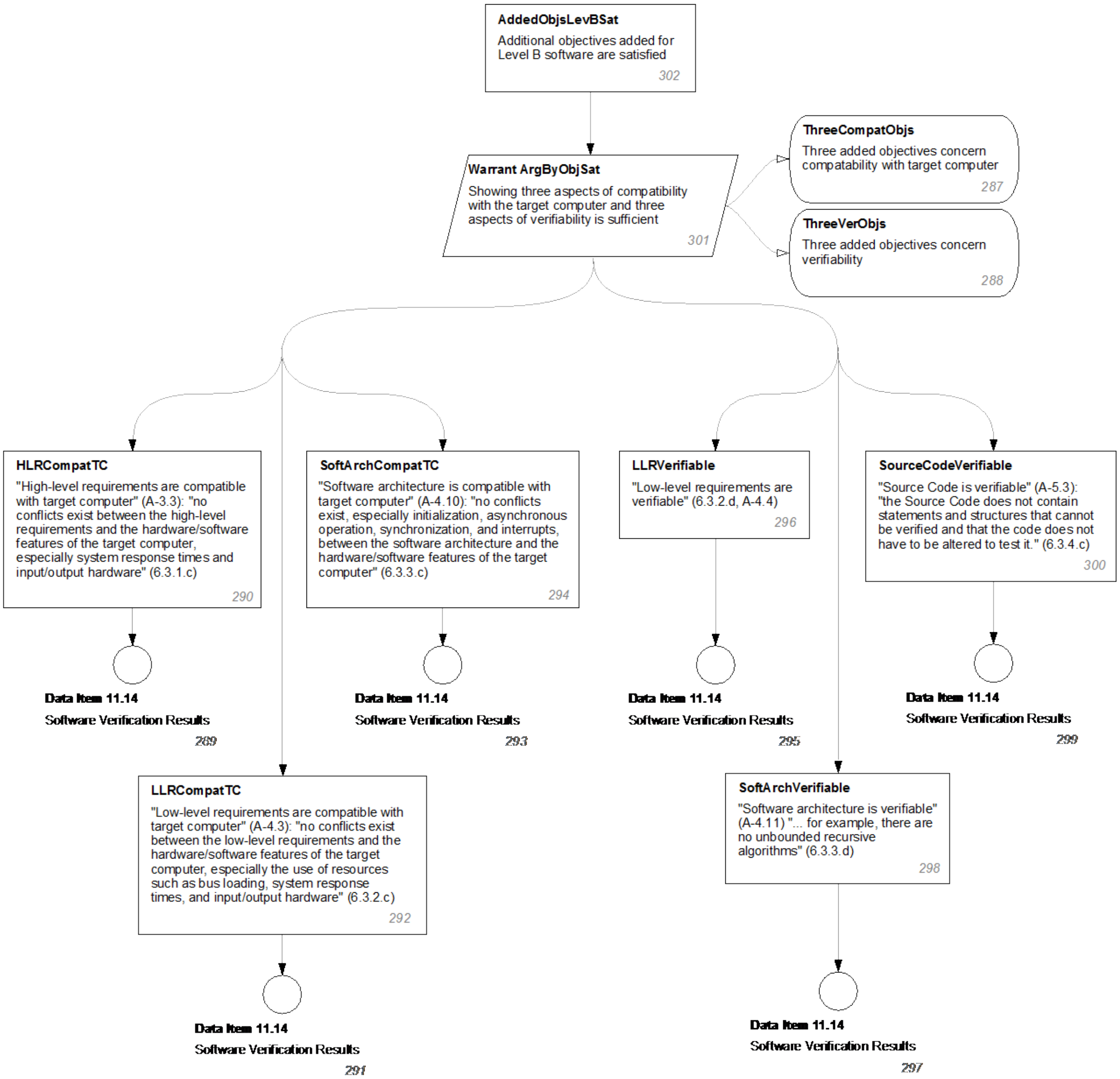

AddedObjsLevBSat
Additional objectives added for Level B software are satisfied
302
Warrant ArgByObjSat
Showing three aspects of compatibility with the target computer and three aspects of verifiability is sufficient
301
ThreeCompatObjs
Three added objectives concern compatability with target computer
287
ThreeVerObjs
Three added objectives concern verifiability
288
HLRCompatTC
"High-level requirements are compatible with target computer" (A-3.3): "no conflicts exist between the high-level requirements and the hardware/software features of the target computer, especially system response times and input/output hardware" (6.3.1.c)
290
SoftArchCompatTC
"Software architecture is compatible with target computer" (A-4.10): "no conflicts exist, especially initialization, asynchronous operation, synchronization, and interrupts, between the software architecture and the hardware/software features of the target computer" (6.3.3.c)
294
LLRVerifiable
"Low-level requirements are verifiable" (6.3.2.d, A-4.4)
296
SourceCodeVerifiable
"Source Code is verifiable" (A-5.3): "the Source Code does not contain statements and structures that cannot be verified and that the code does not have to be altered to test it." (6.3.4.c)
300
Data Item 11.14
Software Verification Results
289
Data Item 11.14
Software Verification Results
293
Data Item 11.14
Software Verification Results
295
Data Item 11.14
Software Verification Results
299
LLRCompatTC
"Low-level requirements are compatible with target computer" (A-4.3): "no conflicts exist between the low-level requirements and the hardware/software features of the target computer, especially the use of resources such as bus loading, system response times, and input/output hardware" (6.3.2.c)
292
SoftArchVerifiable
"Software architecture is verifiable" (A-4.11) "... for example, there are no unbounded recursive algorithms" (6.3.3.d)
298
Data Item 11.14
Software Verification Results
291
Data Item 11.14
Software Verification Results
297

**Figure A-27. Level B: AddedObjsLevBSat**

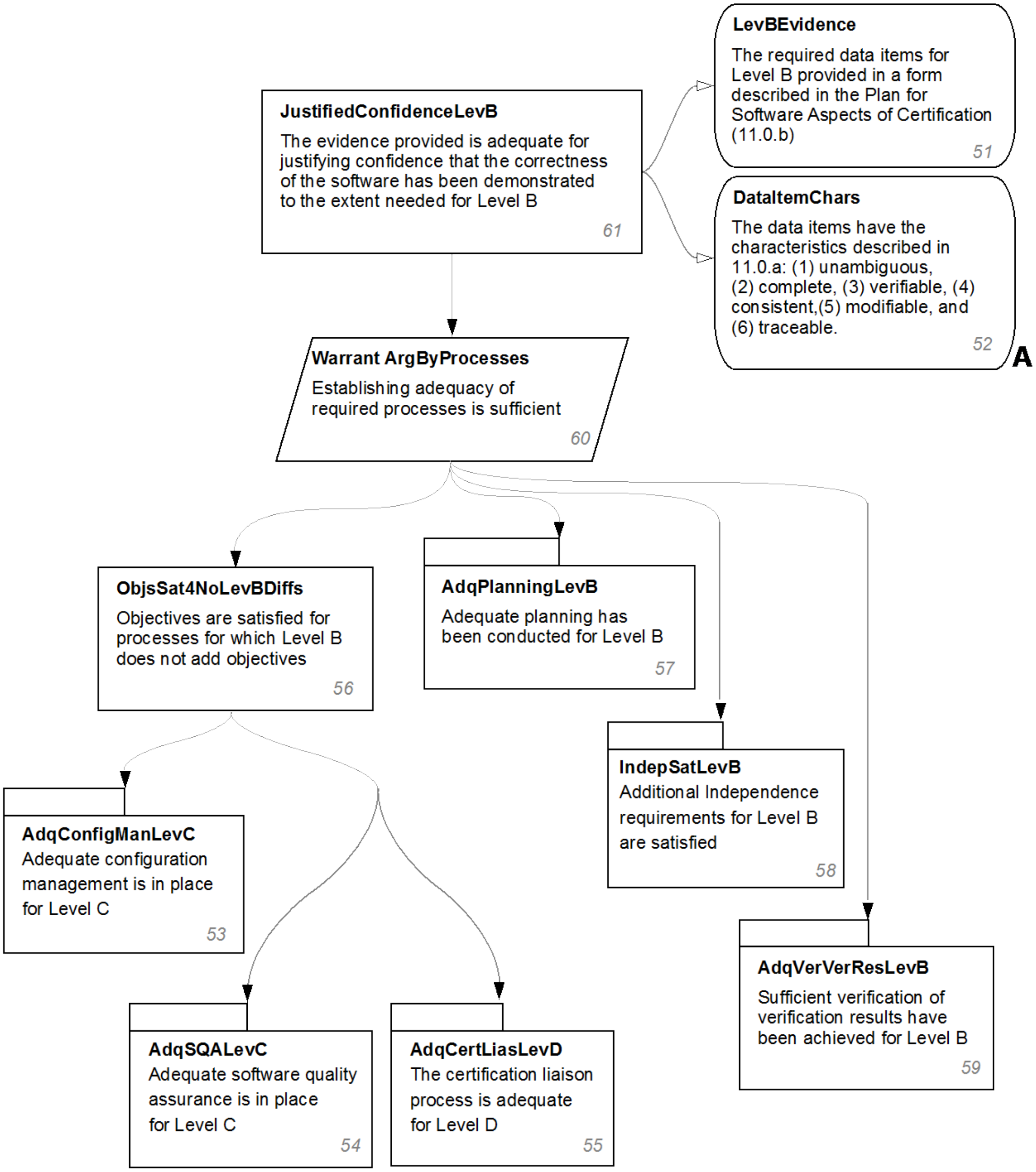

LevBEvidence
The required data items for Level B provided in a form described in the Plan for Software Aspects of Certification (11.0.b)
51
JustifiedConfidenceLevB
The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for Level B
61
DataItemChars
The data items have the characteristics described in 11.0.a: (1) unambiguous, (2) complete, (3) verifiable, (4) consistent,(5) modifiable, and (6) traceable.
52
A
Warrant ArgByProcesses
Establishing adequacy of required processes is sufficient
60
ObjsSat4NoLevBDiffs
Objectives are satisfied for processes for which Level B does not add objectives
56
AdqPlanningLevB
Adequate planning has been conducted for Level B
57
IndepSatLevB
Additional Independence requirements for Level B are satisfied
58
AdqConfigManLevC
Adequate configuration management is in place for Level C
53
AdqVerVerResLevB
Sufficient verification of verification results have been achieved for Level B
59
AdqSQALevC
Adequate software quality assurance is in place for Level C
54
AdqCertLiasLevD
The certification liaison process is adequate for Level D
55

**Figure A-28. Level B: JustifiedConfidenceLevB**

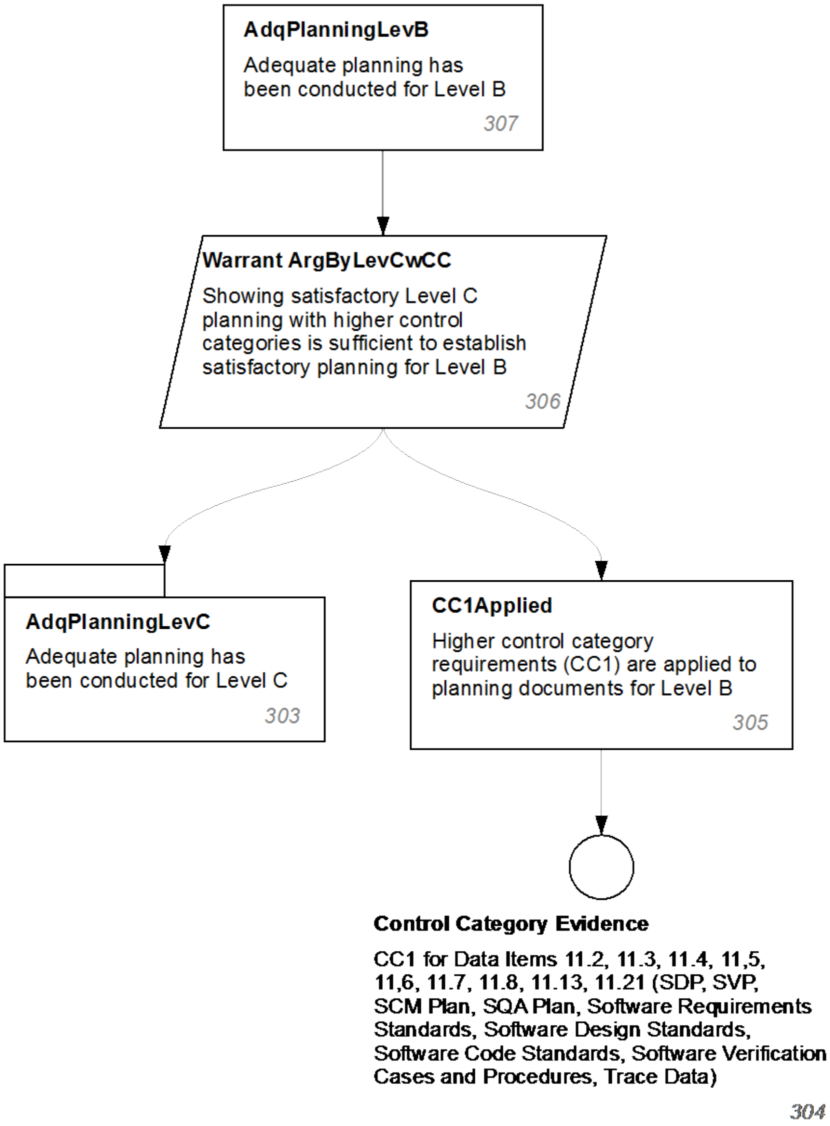

AdqPlanningLevB
Adequate planning has been conducted for Level B
307
Warrant ArgByLevCwCC
Showing satisfactory Level C planning with higher control categories is sufficient to establish satisfactory planning for Level B
306
AdqPlanningLevC
Adequate planning has been conducted for Level C
303
CC1Applied
Higher control category requirements (CC1) are applied to planning documents for Level B
305
Control Category Evidence
CC1 for Data Items 11.2, 11.3, 11.4, 11,5, 11,6, 11.7, 11.8, 11.13, 11.21 (SDP, SVP, SCM Plan, SQA Plan, Software Requirements Standards, Software Design Standards, Software Code Standards, Software Verification Cases and Procedures, Trace Data)
304

**Figure A-29. Level B: AdqPlanningLevB**

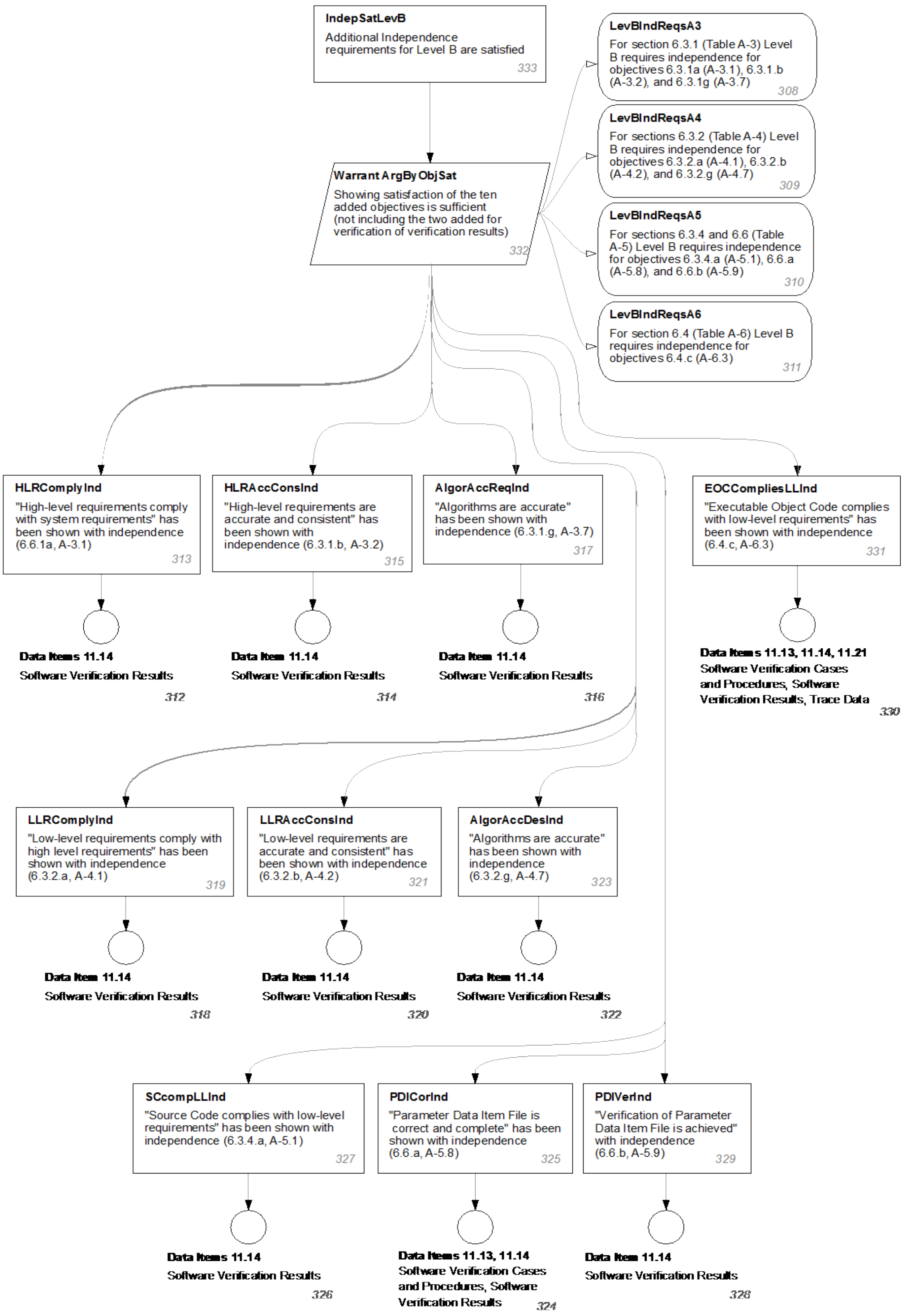

IndepSatLevB
Additional Independence requirements for Level B are satisfied
333
LevBIndReqsA3
For section 6.3.1 (Table A-3) Level B requires independence for objectives 6.3.1a (A-3.1), 6.3.1.b (A-3.2), and 6.3.1g (A-3.7)
308
LevBIndReqsA4
For sections 6.3.2 (Table A-4) Level B requires independence for objectives 6.3.2.a (A-4.1), 6.3.2.b (A-4.2), and 6.3.2.g (A-4.7)
309
Warrant ArgByObjSat
Showing satisfaction of the ten added objectives is sufficient (not including the two added for verification of verification results)
332
LevBIndReqsA5
For sections 6.3.4 and 6.6 (Table A-5) Level B requires independence for objectives 6.3.4.a (A-5.1), 6.6.a (A-5.8), and 6.6.b (A-5.9)
310
LevBIndReqsA6
For section 6.4 (Table A-6) Level B requires independence for objectives 6.4.c (A-6.3)
311
HLRComplyInd
"High-level requirements comply with system requirements" has been shown with independence (6.6.1a, A-3.1)
313
HLRAccConsInd
"High-level requirements are accurate and consistent" has been shown with independence (6.3.1.b, A-3.2)
315
AlgorAccReqInd
"Algorithms are accurate" has been shown with independence (6.3.1.g, A-3.7)
317
EOCCompliesLLInd
"Executable Object Code complies with low-level requirements" has been shown with independence (6.4.c, A-6.3)
331
Data Items 11.14
Software Verification Results
312
Data Item 11.14
Software Verification Results
314
Data Item 11.14
Software Verification Results
316
Data Items 11.13, 11.14, 11.21
Software Verification Cases and Procedures, Software Verification Results, Trace Data
330
LLRComplyInd
"Low-level requirements comply with high level requirements" has been shown with independence (6.3.2.a, A-4.1)
319
LLRAccConsInd
"Low-level requirements are accurate and consistent" has been shown with independence (6.3.2.b, A-4.2)
321
AlgorAccDesInd
"Algorithms are accurate" has been shown with independence (6.3.2.g, A-4.7)
323
Data Item 11.14
Software Verification Results
318
Data Item 11.14
Software Verification Results
320
Data Item 11.14
Software Verification Results
322
SCcompLLInd
"Source Code complies with low-level requirements" has been shown with independence (6.3.4.a, A-5.1)
327
PDICorInd
"Parameter Data Item File is correct and complete" has been shown with independence (6.6.a, A-5.8)
325
PDIVerInd
"Verification of Parameter Data Item File is achieved" with independence (6.6.b, A-5.9)
329
Data Items 11.14
Software Verification Results
326
Data Items 11.13, 11.14
Software Verification Cases and Procedures, Software Verification Results
324
Data Item 11.14
Software Verification Results
328

**Figure A-30. Level B: IndepSatLevB**

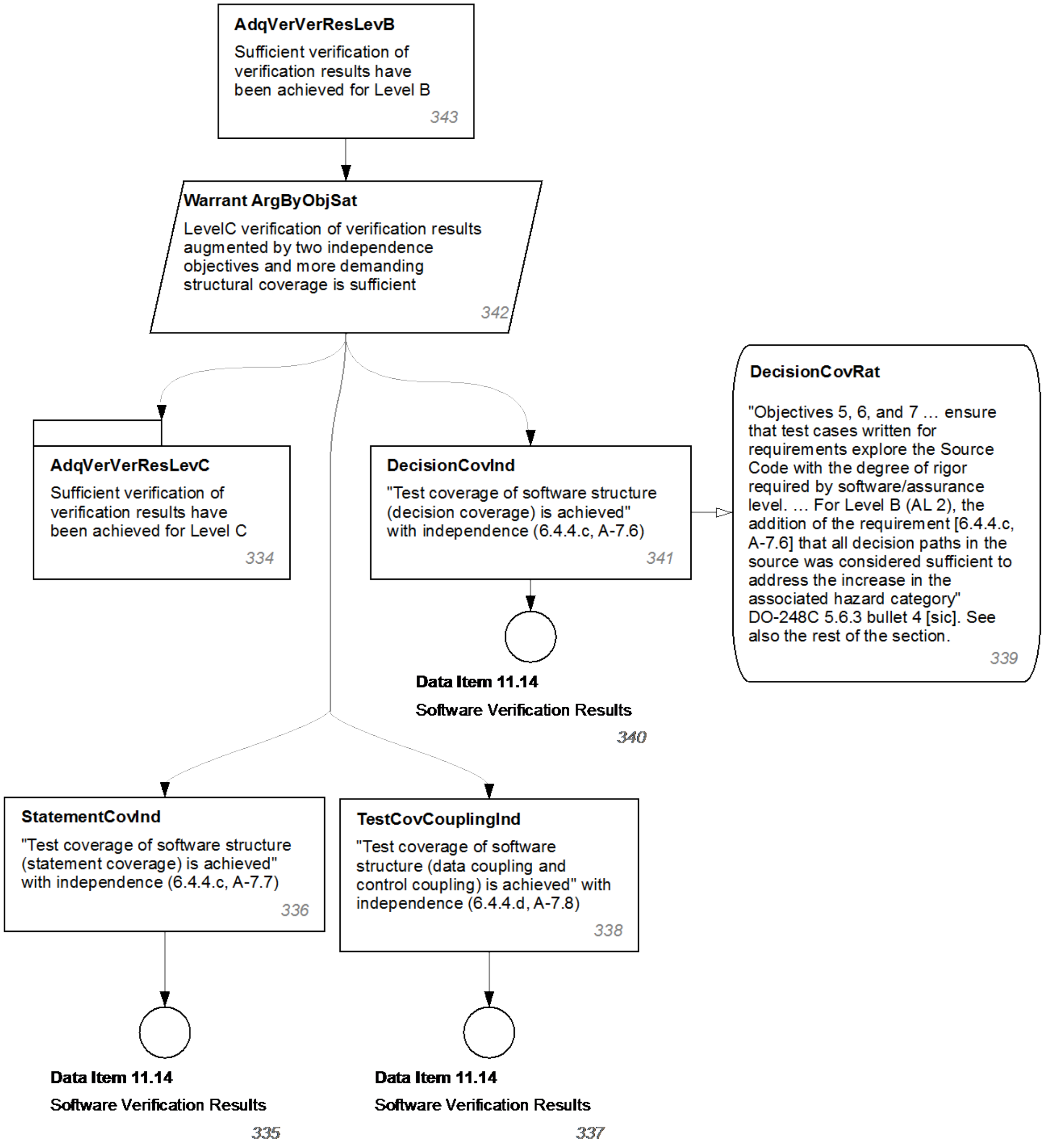

AdqVerVerResLevB
Sufficient verification of verification results have been achieved for Level B
343
Warrant ArgByObjSat
LevelC verification of verification results augmented by two independence objectives and more demanding structural coverage is sufficient
342
AdqVerVerResLevC
Sufficient verification of verification results have been achieved for Level C
334
DecisionCovInd
"Test coverage of software structure (decision coverage) is achieved" with independence (6.4.4.c, A-7.6)
341
DecisionCovRat
"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 B (AL 2), the addition of the requirement [6.4.4.c, A-7.6] that all decision paths in the source was considered sufficient to address the increase in the associated hazard category" DO-248C 5.6.3 bullet 4 [sic]. See also the rest of the section.
339
Data Item 11.14
Software Verification Results
340
StatementCovInd
"Test coverage of software structure (statement coverage) is achieved" with independence (6.4.4.c, A-7.7)
336
TestCovCouplingInd
"Test coverage of software structure (data coupling and control coupling) is achieved" with independence (6.4.4.d, A-7.8)
338
Data Item 11.14
Software Verification Results
335
Data Item 11.14
Software Verification Results
337

**Figure A-31. Level B: AdqVerVerLevB**

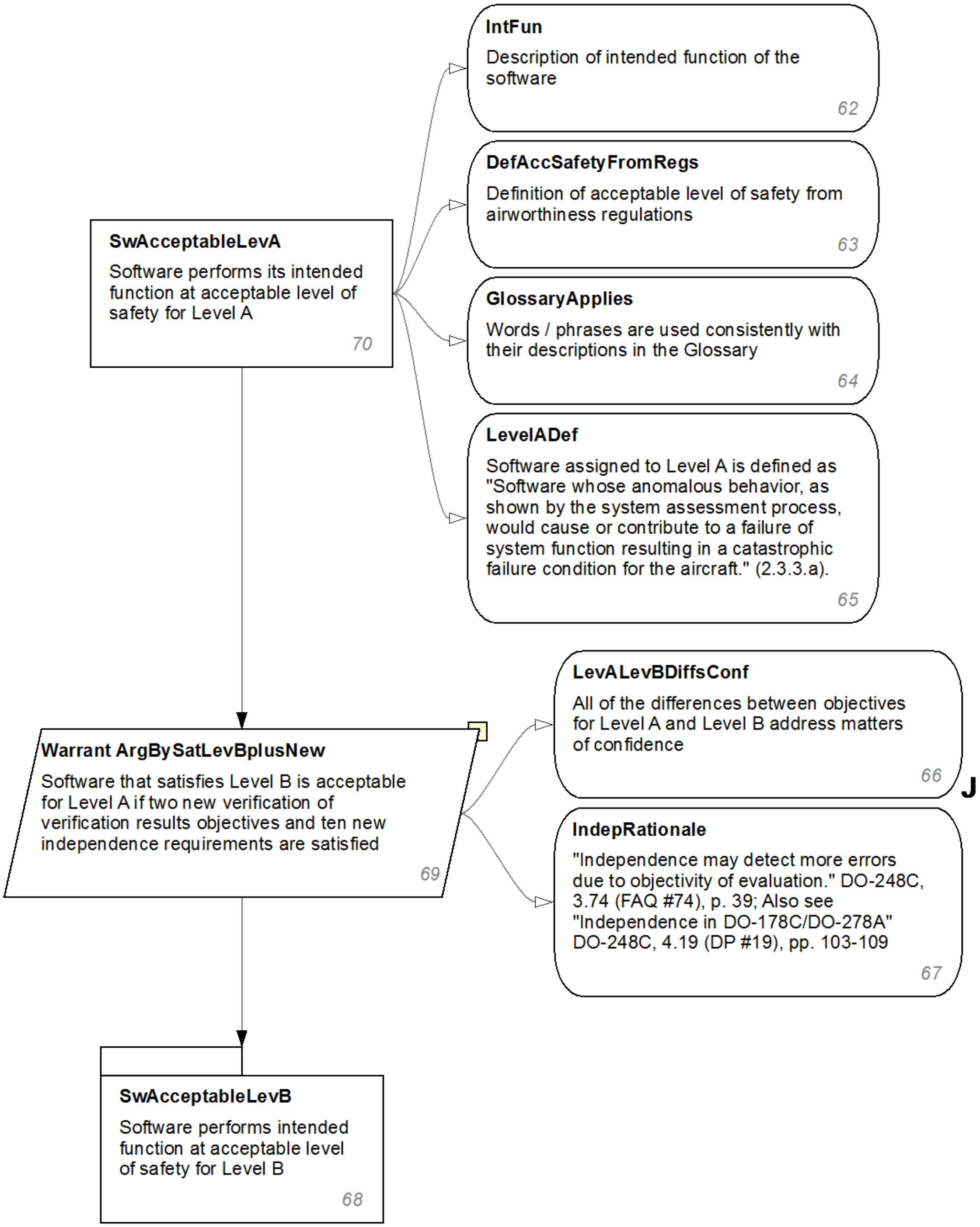

IntFun
Description of intended function of the software
62
DefAccSafetyFromRegs
Definition of acceptable level of safety from airworthiness regulations
63
SwAcceptableLevA
Software performs its intended function at acceptable level of safety for Level A
70
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
64
LevelADef
Software assigned to Level A is defined as "Software whose anomalous behavior, as shown by the system assessment process, would cause or contribute to a failure of system function resulting in a catastrophic failure condition for the aircraft." (2.3.3.a).
65
LevALevBDiffsConf
All of the differences between objectives for Level A and Level B address matters of confidence
66
J
Warrant ArgBySatLevBplusNew
Software that satisfies Level B is acceptable for Level A if two new verification of verification results objectives and ten new independence requirements are satisfied
69
IndepRationale
"Independence may detect more errors due to objectivity of evaluation." DO-248C, 3.74 (FAQ #74), p. 39; Also see "Independence in DO-178C/DO-278A" DO-248C, 4.19 (DP #19), pp. 103-109
67
SwAcceptableLevB
Software performs intended function at acceptable level of safety for Level B
68

**Figure A-32. Level A: SwAcceptableLevA**

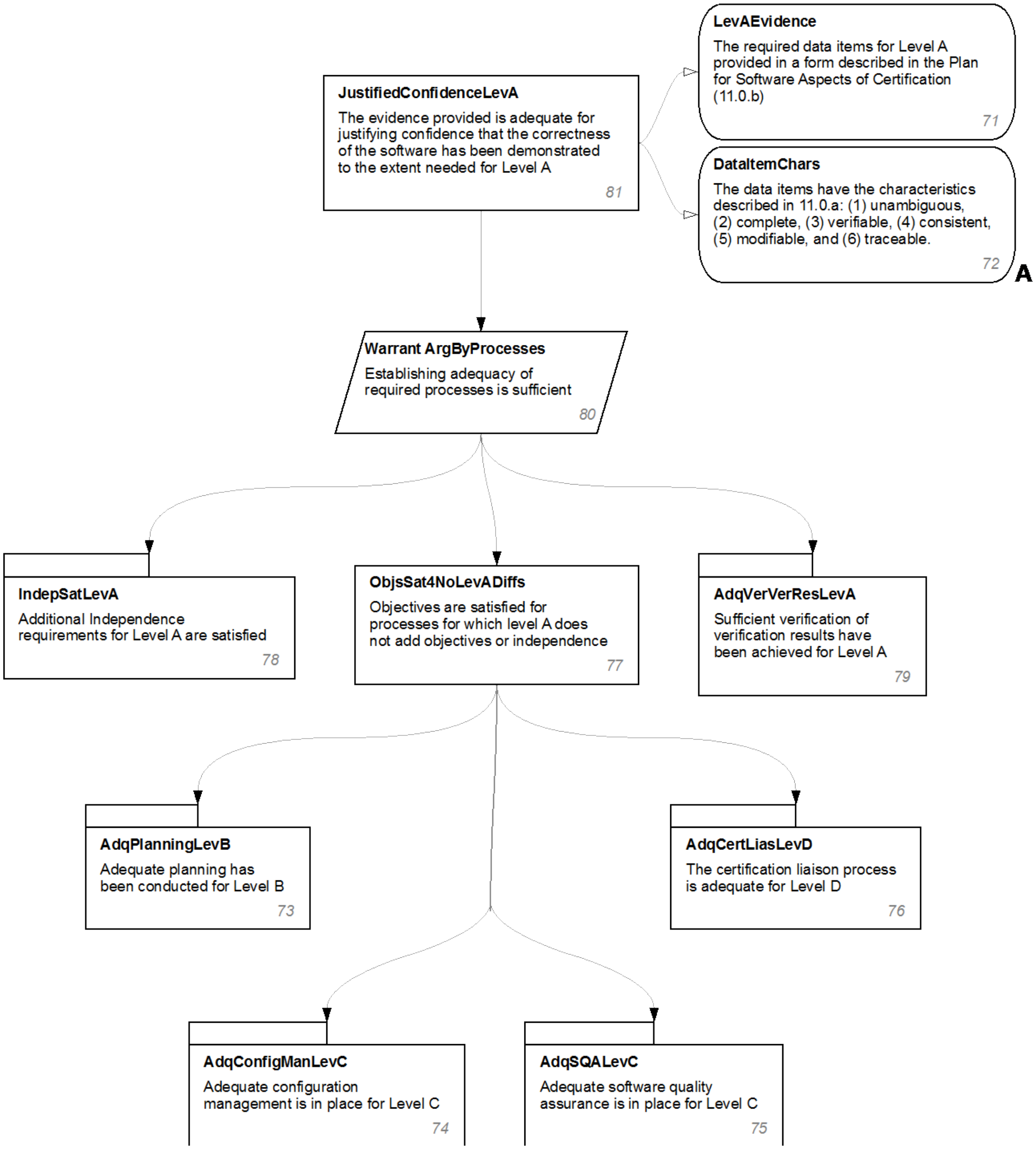

JustifiedConfidenceLevA
The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for Level A
81
LevAEvidence
The required data items for Level A provided in a form described in the Plan for Software Aspects of Certification (11.0.b)
71
DataItemChars
The data items have the characteristics described in 11.0.a: (1) unambiguous, (2) complete, (3) verifiable, (4) consistent, (5) modifiable, and (6) traceable.
72
A
Warrant ArgByProcesses
Establishing adequacy of required processes is sufficient
80
IndepSatLevA
Additional Independence requirements for Level A are satisfied
78
ObjsSat4NoLevADiffs
Objectives are satisfied for processes for which level A does not add objectives or independence
77
AdqVerVerResLevA
Sufficient verification of verification results have been achieved for Level A
79
AdqPlanningLevB
Adequate planning has been conducted for Level B
73
AdqCertLiasLevD
The certification liaison process is adequate for Level D
76
AdqConfigManLevC
Adequate configuration management is in place for Level C
74
AdqSQALevC
Adequate software quality assurance is in place for Level C
75

**Figure A-33. Level A: JustifiedConfidenceLevA**

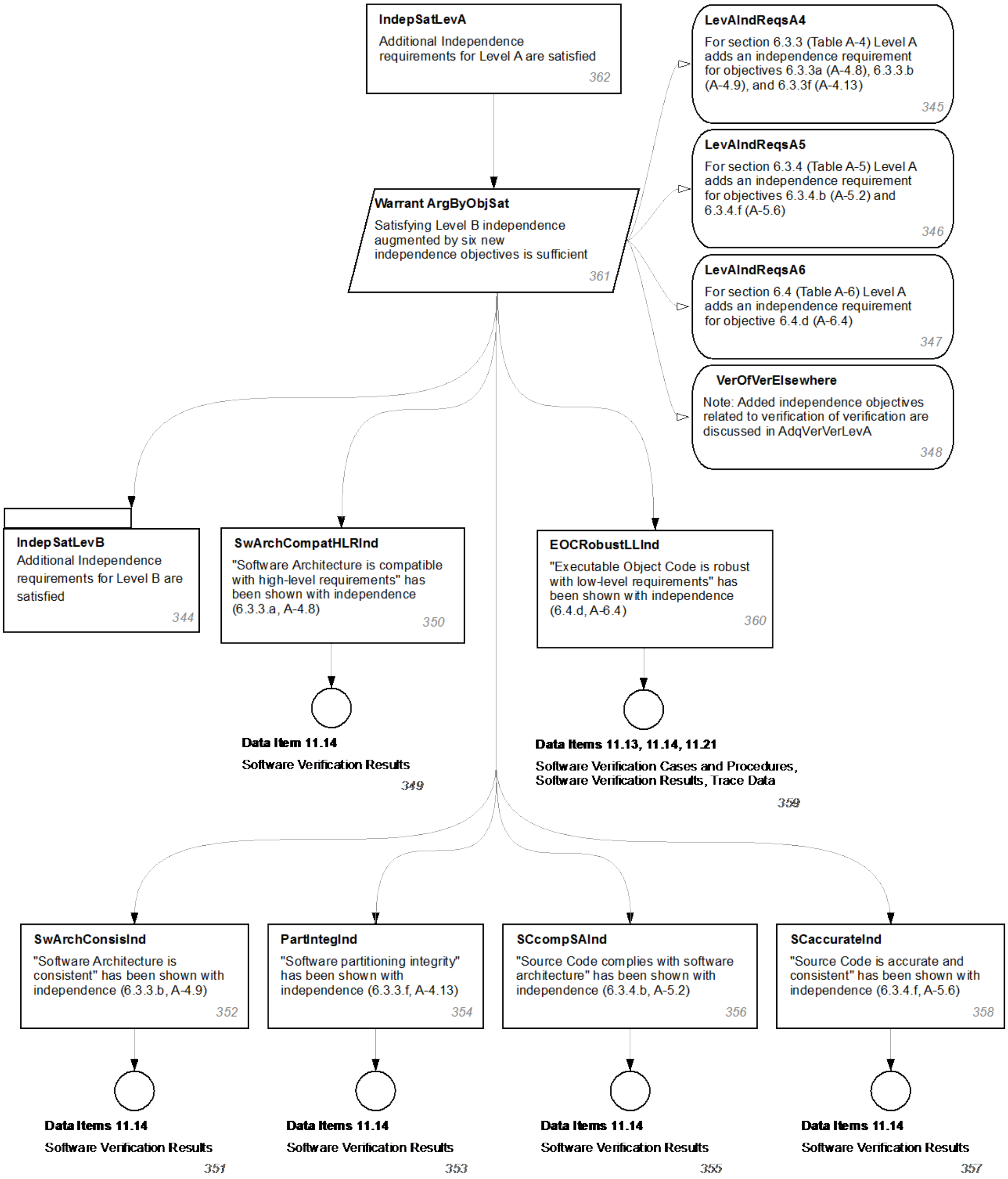

IndepSatLevA
Additional Independence requirements for Level A are satisfied
362
LevAIndReqsA4
For section 6.3.3 (Table A-4) Level A adds an independence requirement for objectives 6.3.3a (A-4.8), 6.3.3.b (A-4.9), and 6.3.3f (A-4.13)
345
LevAIndReqsA5
For section 6.3.4 (Table A-5) Level A adds an independence requirement for objectives 6.3.4.b (A-5.2) and 6.3.4.f (A-5.6)
346
Warrant ArgByObjSat
Satisfying Level B independence augmented by six new independence objectives is sufficient
361
LevAIndReqsA6
For section 6.4 (Table A-6) Level A adds an independence requirement for objective 6.4.d (A-6.4)
347
VerOfVerElsewhere
Note: Added independence objectives related to verification of verification are discussed in AdqVerVerLevA
348
IndepSatLevB
Additional Independence requirements for Level B are satisfied
344
SwArchCompatHLRInd
"Software Architecture is compatible with high-level requirements" has been shown with independence (6.3.3.a, A-4.8)
350
EOCRobustLLInd
"Executable Object Code is robust with low-level requirements" has been shown with independence (6.4.d, A-6.4)
360
Data Item 11.14
Software Verification Results
349
Data Items 11.13, 11.14, 11.21
Software Verification Cases and Procedures, Software Verification Results, Trace Data
359
SwArchConsisInd
"Software Architecture is consistent" has been shown with independence (6.3.3.b, A-4.9)
352
PartIntegInd
"Software partitioning integrity" has been shown with independence (6.3.3.f, A-4.13)
354
SCcompSAInd
"Source Code complies with software architecture" has been shown with independence (6.3.4.b, A-5.2)
356
SCaccurateInd
"Source Code is accurate and consistent" has been shown with independence (6.3.4.f, A-5.6)
358
Data Items 11.14
Software Verification Results
351
Data Items 11.14
Software Verification Results
353
Data Items 11.14
Software Verification Results
355
Data Items 11.14
Software Verification Results
357

**Figure A-34. Level A: IndepSatLevA**

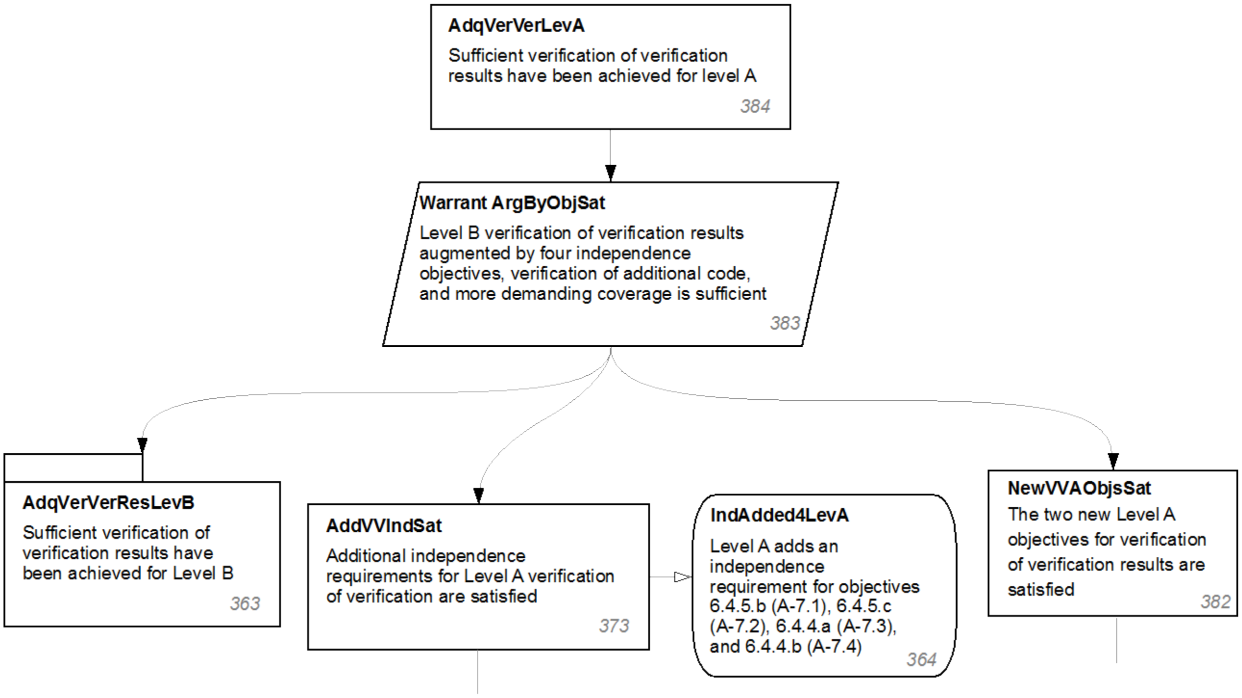

AdqVerVerLevA
Sufficient verification of verification results have been achieved for level A
384
Warrant ArgByObjSat
Level B verification of verification results augmented by four independence objectives, verification of additional code, and more demanding coverage is sufficient
383
AdqVerVerResLevB
Sufficient verification of verification results have been achieved for Level B
363
AddVVIndSat
Additional independence requirements for Level A verification of verification are satisfied
373
IndAdded4LevA
Level A adds an independence requirement for objectives 6.4.5.b (A-7.1), 6.4.5.c (A-7.2), 6.4.4.a (A-7.3), and 6.4.4.b (A-7.4)
364
NewVVAObjsSat
The two new Level A objectives for verification of verification results are satisfied
382

**Figure A-35. Level A: AdqVerVerLevA (top)**

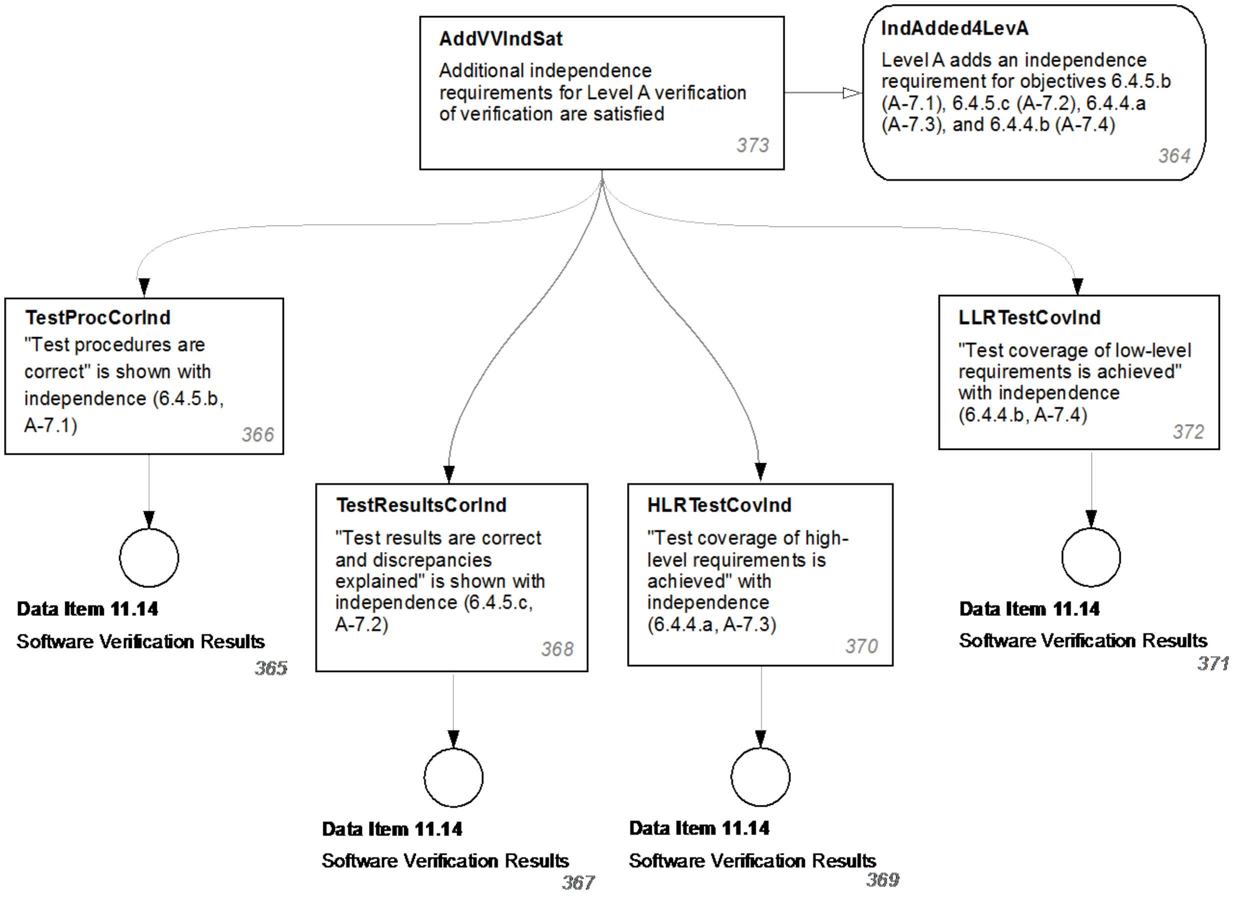

AddVVIndSat
Additional independence requirements for Level A verification of verification are satisfied
373
IndAdded4LevA
Level A adds an independence requirement for objectives 6.4.5.b (A-7.1), 6.4.5.c (A-7.2), 6.4.4.a (A-7.3), and 6.4.4.b (A-7.4)
364
TestProcCorInd
"Test procedures are correct" is shown with independence (6.4.5.b, A-7.1)
366
TestResultsCorInd
"Test results are correct and discrepancies explained" is shown with independence (6.4.5.c, A-7.2)
368
HLRTestCovInd
"Test coverage of high-level requirements is achieved" with independence (6.4.4.a, A-7.3)
370
LLRTestCovInd
"Test coverage of low-level requirements is achieved" with independence (6.4.4.b, A-7.4)
372
Data Item 11.14
Software Verification Results
365
Data Item 11.14
Software Verification Results
367
Data Item 11.14
Software Verification Results
369
Data Item 11.14
Software Verification Results
371

**Figure A-36. Level A: AdqVerVerLevA (left)**

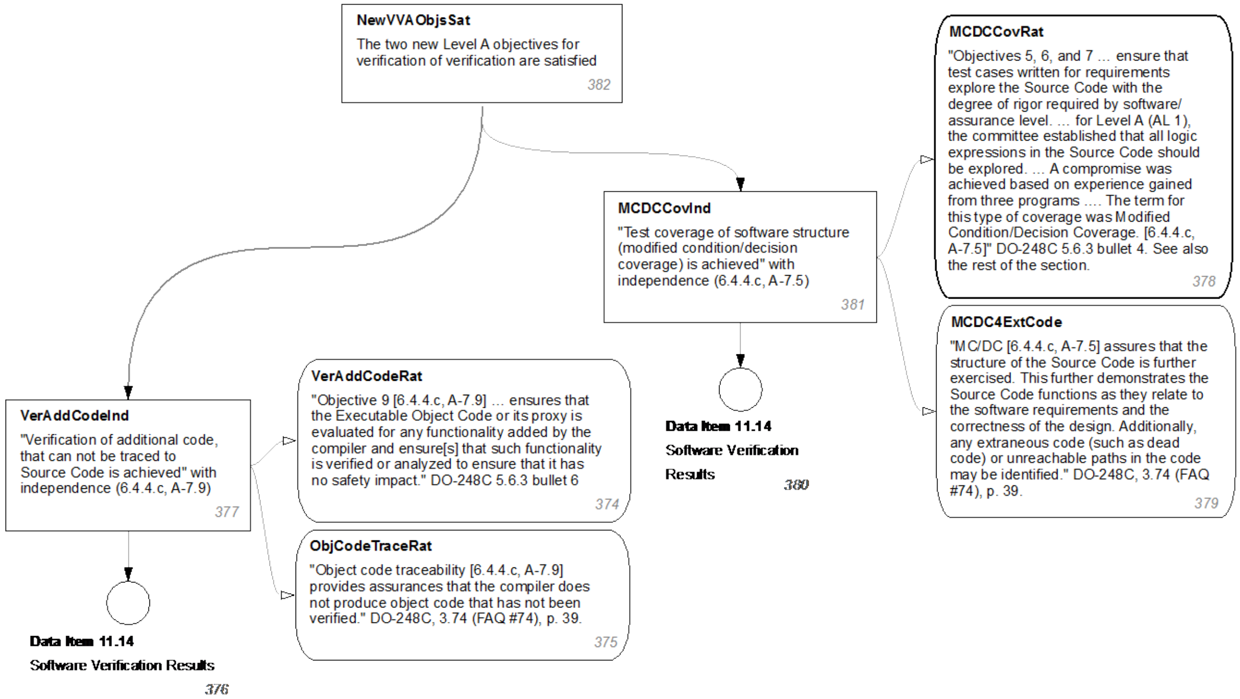

NewVVAObjsSat
The two new Level A objectives for verification of verification are satisfied
382
MCDCCovRat
"Objectives 5, 6, and 7 ... ensure that test cases written for requirements explore the Source Code with the degree of rigor required by software/ assurance level. ... for Level A (AL 1), the committee 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. [6.4.4.c, A-7.5]" DO-248C 5.6.3 bullet 4. See also the rest of the section.
378
MCDCCovInd
"Test coverage of software structure (modified condition/decision coverage) is achieved" with independence (6.4.4.c, A-7.5)
381
MCDC4ExtCode
"MC/DC [6.4.4.c, A-7.5] 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." DO-248C, 3.74 (FAQ #74), p. 39.
379
Data Item 11.14
Software Verification Results
380
VerAddCodeRat
"Objective 9 [6.4.4.c, A-7.9] ... 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
374
VerAddCodeInd
"Verification of additional code, that can not be traced to Source Code is achieved" with independence (6.4.4.c, A-7.9)
377
ObjCodeTraceRat
"Object code traceability [6.4.4.c, A-7.9] provides assurances that the compiler does not produce object code that has not been verified." DO-248C, 3.74 (FAQ #74), p. 39.
375
Data Item 11.14
Software Verification Results
376

**Figure A-37. Level A: ADQVERVERLEVA (right)**

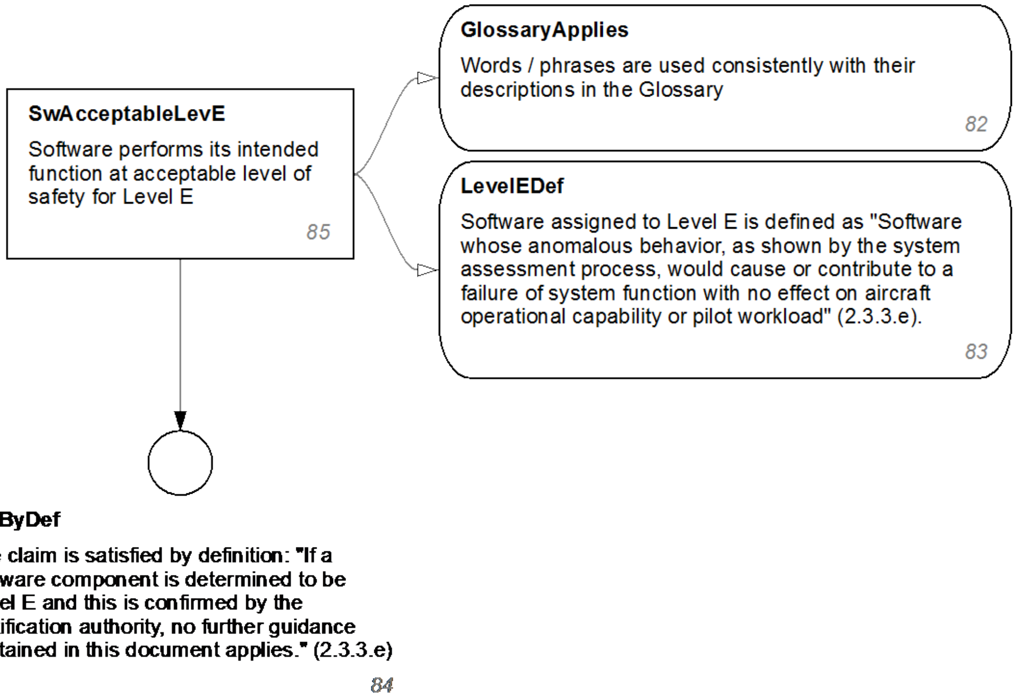

GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
82
SwAcceptableLevE
Software performs its intended function at acceptable level of safety for Level E
85
LevelEDef
Software assigned to Level E is defined as "Software whose anomalous behavior, as shown by the system assessment process, would cause or contribute to a failure of system function with no effect on aircraft operational capability or pilot workload" (2.3.3.e).
83
SatByDef
The claim is satisfied by definition: "If a software component is determined to be Level E and this is confirmed by the certification authority, no further guidance contained in this document applies." (2.3.3.e)
84

**Figure A-38. Level E: SWACCEPTABLELEVE**

# APPENDIX B—ARGUMENTS FOR TOOL QUALIFICATION (DO-330)

This appendix contains the arguments developed for tool qualification in accordance with DO-178C §12.2 and DO-330. Section B.1. identifies differences between the DO-330 and DO-178C certification contexts that are critical to understanding how and why the DO-330 argument differs from the DO-178C argument. Section B.2. presents the top level of the tool qualification argument, which corresponds to DO-178C §12.2. Section B.3. presents the remainder of the argument for TQL-5 tools. Subsequent sections present the remainder of the arguments for tools at higher Tool Qualification Levels.

## B.1. THE CERTIFICATION CONTEXT FOR DO-330

Although the structure of DO-330 is largely analogous to DO-178C, there are differences that affect the structure of the tool-qualification argument. Figure B-1 illustrates two key differences: 1) the source of the highest level of requirements; and 2) how safety is managed and safety-related insights and changes are handled.

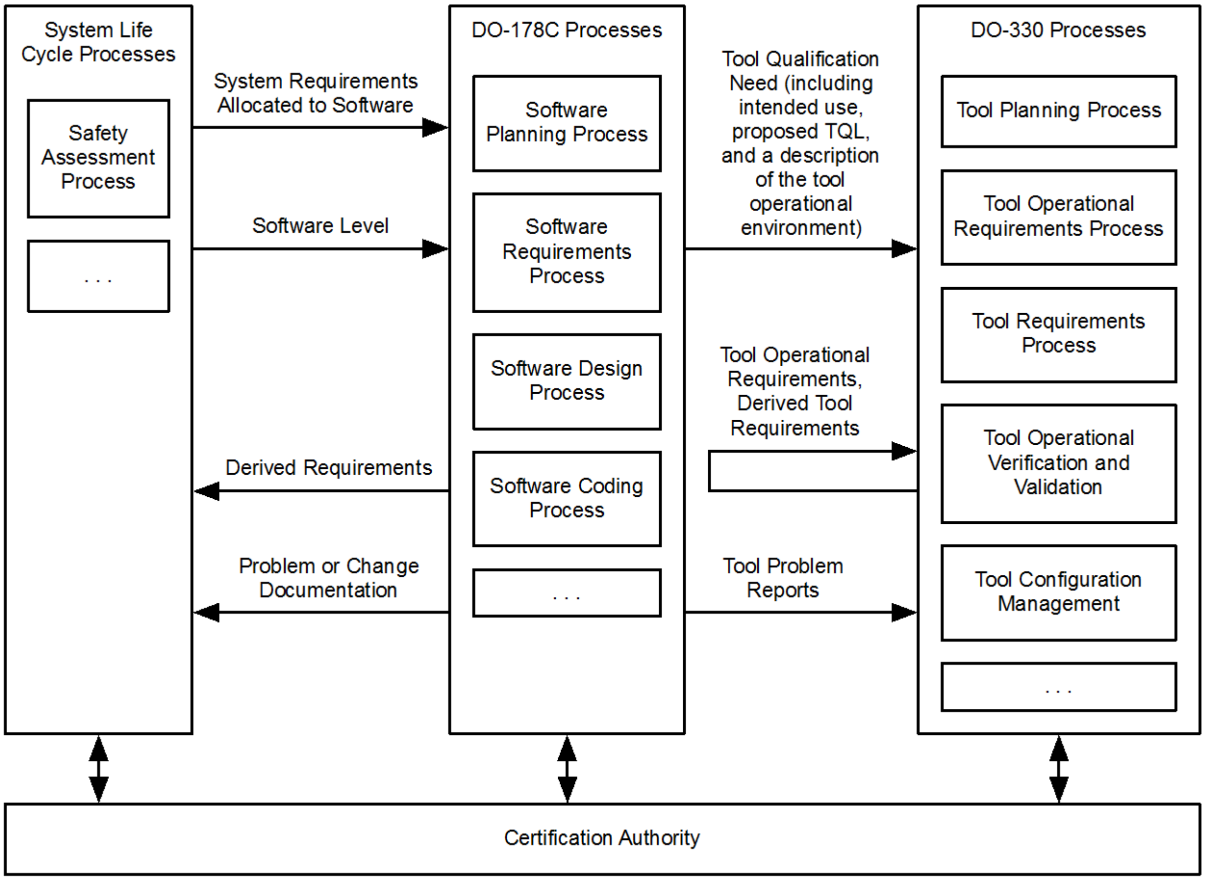

System Life Cycle Processes
Safety Assessment Process
. . .
System Requirements Allocated to Software
Software Level
Derived Requirements
Problem or Change Documentation
DO-178C Processes
Software Planning Process
Software Requirements Process
Software Design Process
Software Coding Process
. . .
Tool Qualification Need (including intended use, proposed TQL, and a description of the tool operational environment)
Tool Operational Requirements, Derived Tool Requirements
Tool Problem Reports
DO-330 Processes
Tool Planning Process
Tool Operational Requirements Process
Tool Requirements Process
Tool Operational Verification and Validation
Tool Configuration Management
. . .
Certification Authority

**Figure B-1. Certification Context for DO-178C and DO-330**

DO-178C assumes that system life-cycle processes have defined system requirements, allocated some of these to software, and determined the appropriate software level. When DO-178C and DO-330 are used together, the DO-178C software planning process defines the need for tool qualification by identifying the tool and its intended use in the airborne software development process. The software-planning process determines the appropriate TQL in accordance with DO-178C §12.2. Tool operational requirements are defined in accordance with the DO-330 tool operational requirements process.

DO-178C assumes the existence of an unspecified system-safety assessment process. Software developers provide derived requirements, problem documentation, and change documentation to the safety assessment process, which updates the system requirements and software level as needed. In contrast, tool requirements and derived tool requirements are produced in accordance with the DO-330 tool requirements process. The same process also justifies the existence of derived tool requirements and evaluates them "to ensure that they do not negatively impact the expected functionality and outputs defined in the Tool Operational Requirements" (DO-330, §5.2.1.2.h).

## B.2. TOP LEVEL (DO-178C §12.2)

The arguments developed for tool qualification begin by showing that the tools used to produce airborne software "provide confidence at least equivalent to that of [sic] the processes eliminated, reduced or automated" (DO-178C, §12.2.1). Figure B-2 illustrates this argument (for airborne software Level D).[1] The conclusion is supported by an argument over the DO-178C tool-qualification process (§12.2). Premises GTQLevDC1 and GTQLevDC23 represent the logic implicit in DO-178C's §12.2.2, which specifies the appropriate TQL based on the airborne software level and three criteria:

- *Criterion 1.* "A tool whose output is part of the airborne software and thus could insert an error."
- *Criterion 2.* "A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of … verification process(es) other than that automated by the tool, or … development process(es) that could have an impact on the airborne software."
- *Criterion 3.* "A too that, within the scope of its intended use, could fail to detect an error."

For airborne software Level D, DO-178C's table 12-1 specifies that tools satisfying criterion 1 must be qualified to TQL-4, whereas tools that satisfy only criterion 2 or criterion 3 need only be qualified to TQL-5.

[1] There are two differences between the style of the DO-330 arguments and the DO-178C arguments. The DO-330 arguments were developed using a simple graphics program instead of the Dependable Computing tools. Also, the DO-330 arguments use the traditional GSN Strategy convention instead of the Warrant approach developed for the DO-178C arguments. Changing to the Warrant approach and reproducing new diagrams may be done if desired.

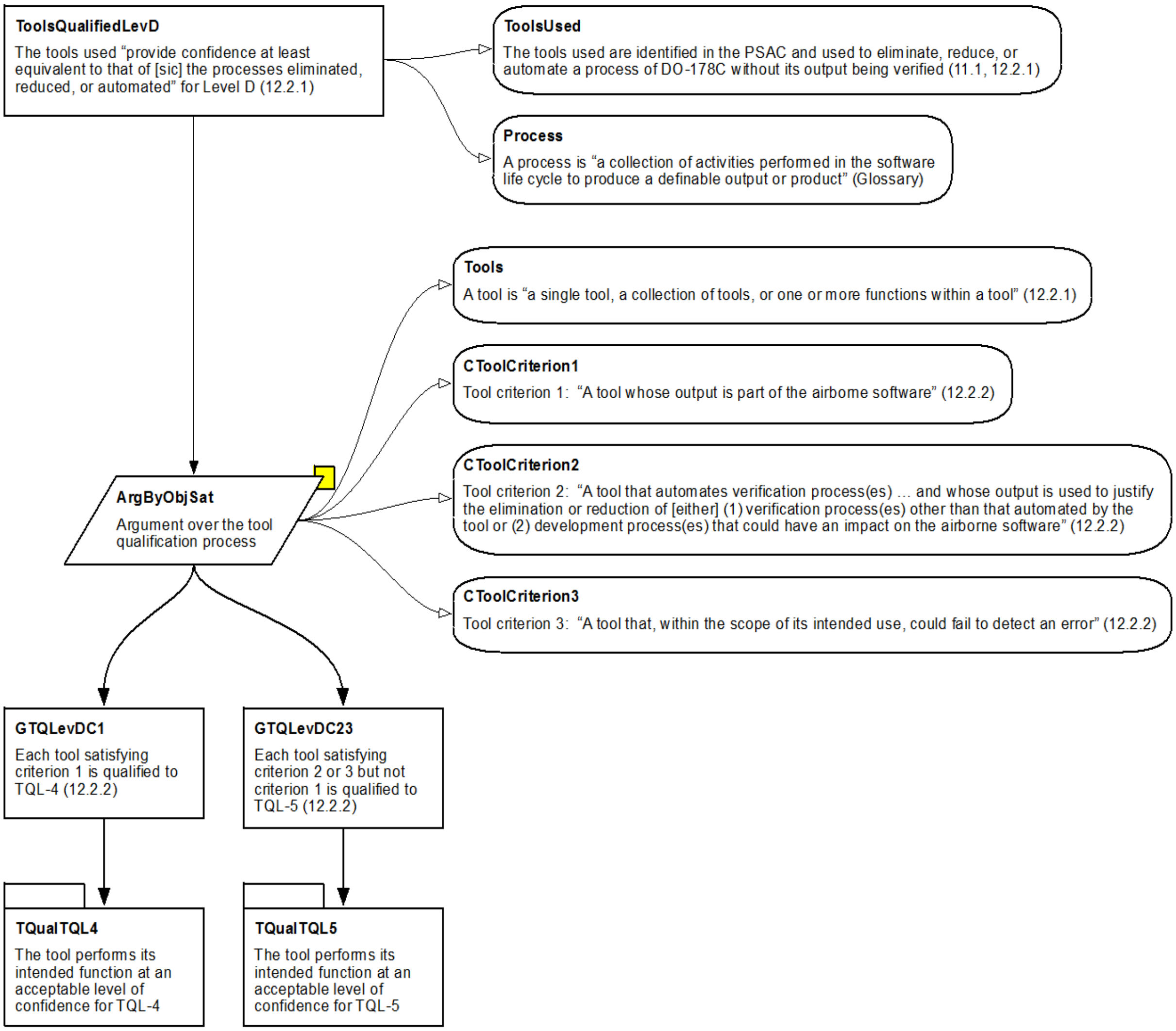

ToolsQualifiedLevD
The tools used "provide confidence at least equivalent to that of [sic] the processes eliminated, reduced, or automated" for Level D (12.2.1)
ToolsUsed
The tools used are identified in the PSAC and used to eliminate, reduce, or automate a process of DO-178C without its output being verified (11.1, 12.2.1)
Process
A process is "a collection of activities performed in the software life cycle to produce a definable output or product" (Glossary)
Tools
A tool is "a single tool, a collection of tools, or one or more functions within a tool" (12.2.1)
CToolCriterion1
Tool criterion 1: "A tool whose output is part of the airborne software" (12.2.2)
CToolCriterion2
Tool criterion 2: "A tool that automates verification process(es) … and whose output is used to justify the elimination or reduction of [either] (1) verification process(es) other than that automated by the tool or (2) development process(es) that could have an impact on the airborne software" (12.2.2)
CToolCriterion3
Tool criterion 3: "A tool that, within the scope of its intended use, could fail to detect an error" (12.2.2)
ArgByObjSat
Argument over the tool qualification process
GTQLevDC1
Each tool satisfying criterion 1 is qualified to TQL-4 (12.2.2)
GTQLevDC23
Each tool satisfying criterion 2 or 3 but not criterion 1 is qualified to TQL-5 (12.2.2)
TQualTQL4
The tool performs its intended function at an acceptable level of confidence for TQL-4
TQualTQL5
The tool performs its intended function at an acceptable level of confidence for TQL-5

**Figure B-2. ToolsQualifedLevD**

Confidence in the argument depicted in figure B-2 might be lost if the tool-qualification needs were identified incorrectly. Figure B-3 depicts the conference argument associated with the inference of ToolsQualifiedLevD from GTQLevDC1 and GTQLevDC23. Support for the conclusion that "there is sufficient confidence that tool qualification needs have been established correctly" is given by evidence of satisfaction of DO-178C objective 4.1.D/A-1.4. That is, the Plan for Software Aspects of Certification (PSAC), Software Development Plan (DSP), Software Verification Plan (SVP), Software Configuration Management (SCM) Plan, and Software Quality Assurance (SQA) Plan testify that "additional considerations," including tool qualification, "have been addressed" (DO-178C, §4.1.D).

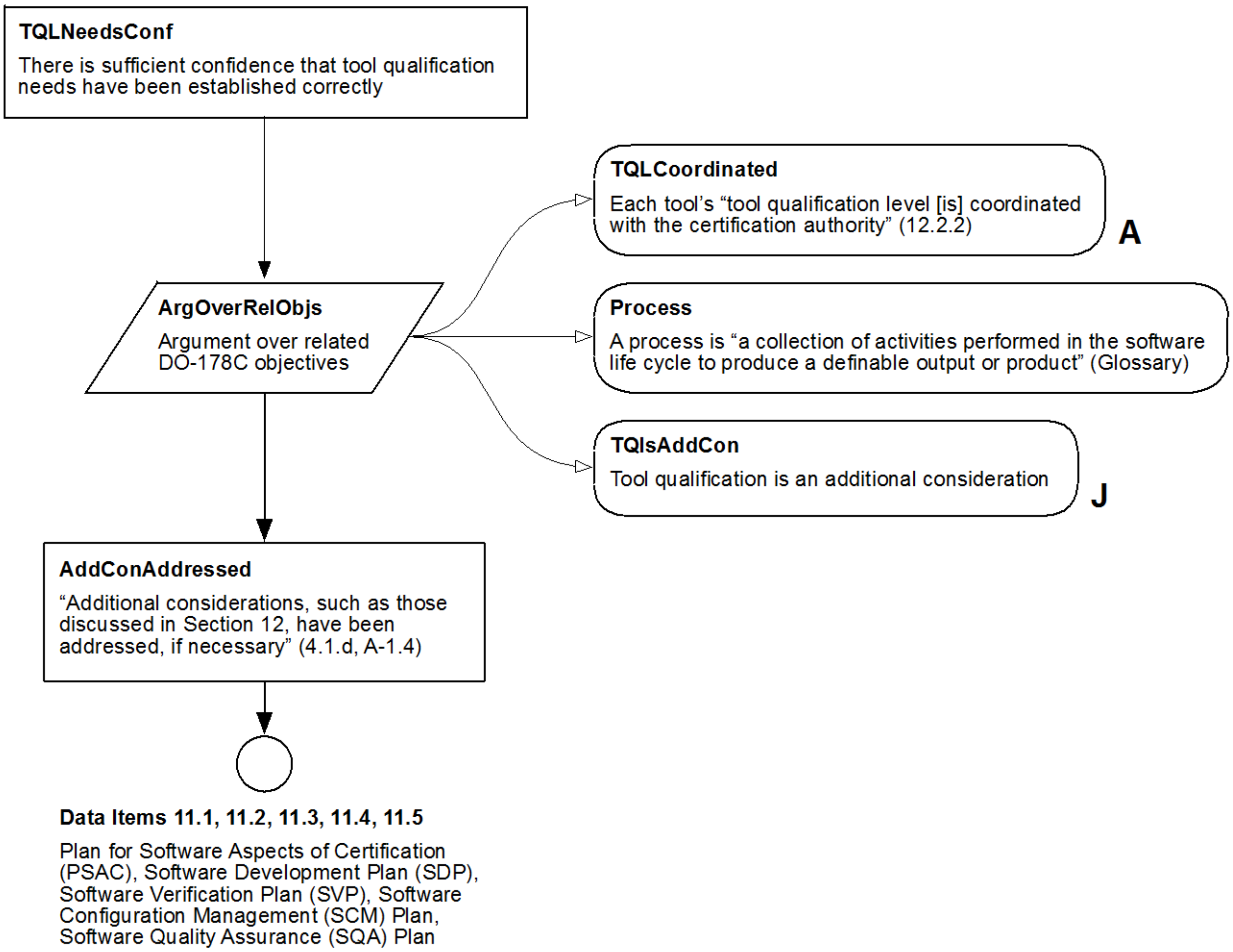

TQLNeedsConf
There is sufficient confidence that tool qualification needs have been established correctly
TQLCoordinated
Each tool's "tool qualification level [is] coordinated with the certification authority" (12.2.2)
A
ArgOverRelObjs
Argument over related DO-178C objectives
Process
A process is "a collection of activities performed in the software life cycle to produce a definable output or product" (Glossary)
TQIsAddCon
Tool qualification is an additional consideration
J
AddConAddressed
"Additional considerations, such as those discussed in Section 12, have been addressed, if necessary" (4.1.d, A-1.4)
Data Items 11.1, 11.2, 11.3, 11.4, 11.5
Plan for Software Aspects of Certification (PSAC), Software Development Plan (SDP), Software Verification Plan (SVP), Software Configuration Management (SCM) Plan, Software Quality Assurance (SQA) Plan

**Figure B-3. TQLNeedsConf**

Three variants of the argument shown in figure B-2 present alternatives that apply at other airborne software levels. Figure B-4 gives the top level of the argument for Level C airborne software. Figure B-5 gives the top level of the argument for Level B airborne software. And figure B-6 gives the top level of the argument for Level A airborne software. Tool qualification is not applicable for Level E airborne software.

All of these arguments depend on evidence that the tool performs its intended function at a level of confidence that is acceptable given its TQL. The confidence argument depicted in figure B-3 applies in all cases. These arguments differ only in that different TQL levels apply at each of the airborne software levels.

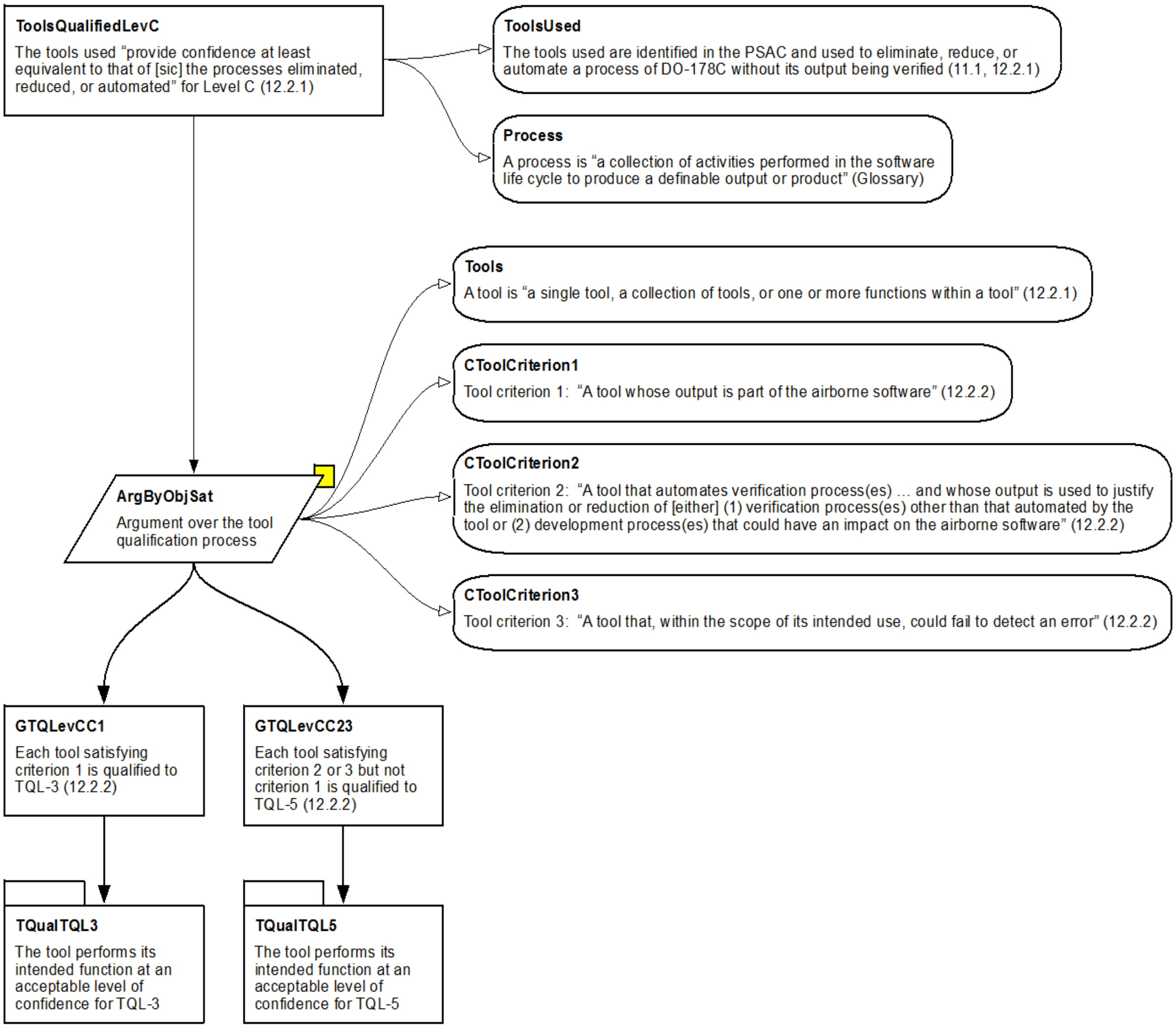

ToolsQualifiedLevC
The tools used "provide confidence at least equivalent to that of [sic] the processes eliminated, reduced, or automated" for Level C (12.2.1)
ToolsUsed
The tools used are identified in the PSAC and used to eliminate, reduce, or automate a process of DO-178C without its output being verified (11.1, 12.2.1)
Process
A process is "a collection of activities performed in the software life cycle to produce a definable output or product" (Glossary)
Tools
A tool is "a single tool, a collection of tools, or one or more functions within a tool" (12.2.1)
CToolCriterion1
Tool criterion 1: "A tool whose output is part of the airborne software" (12.2.2)
CToolCriterion2
Tool criterion 2: "A tool that automates verification process(es) ... and whose output is used to justify the elimination or reduction of [either] (1) verification process(es) other than that automated by the tool or (2) development process(es) that could have an impact on the airborne software" (12.2.2)
CToolCriterion3
Tool criterion 3: "A tool that, within the scope of its intended use, could fail to detect an error" (12.2.2)
ArgByObjSat
Argument over the tool qualification process
GTQLevCC1
Each tool satisfying criterion 1 is qualified to TQL-3 (12.2.2)
GTQLevCC23
Each tool satisfying criterion 2 or 3 but not criterion 1 is qualified to TQL-5 (12.2.2)
TQualTQL3
The tool performs its intended function at an acceptable level of confidence for TQL-3
TQualTQL5
The tool performs its intended function at an acceptable level of confidence for TQL-5

**Figure B-4. ToolsQualifedLevC**

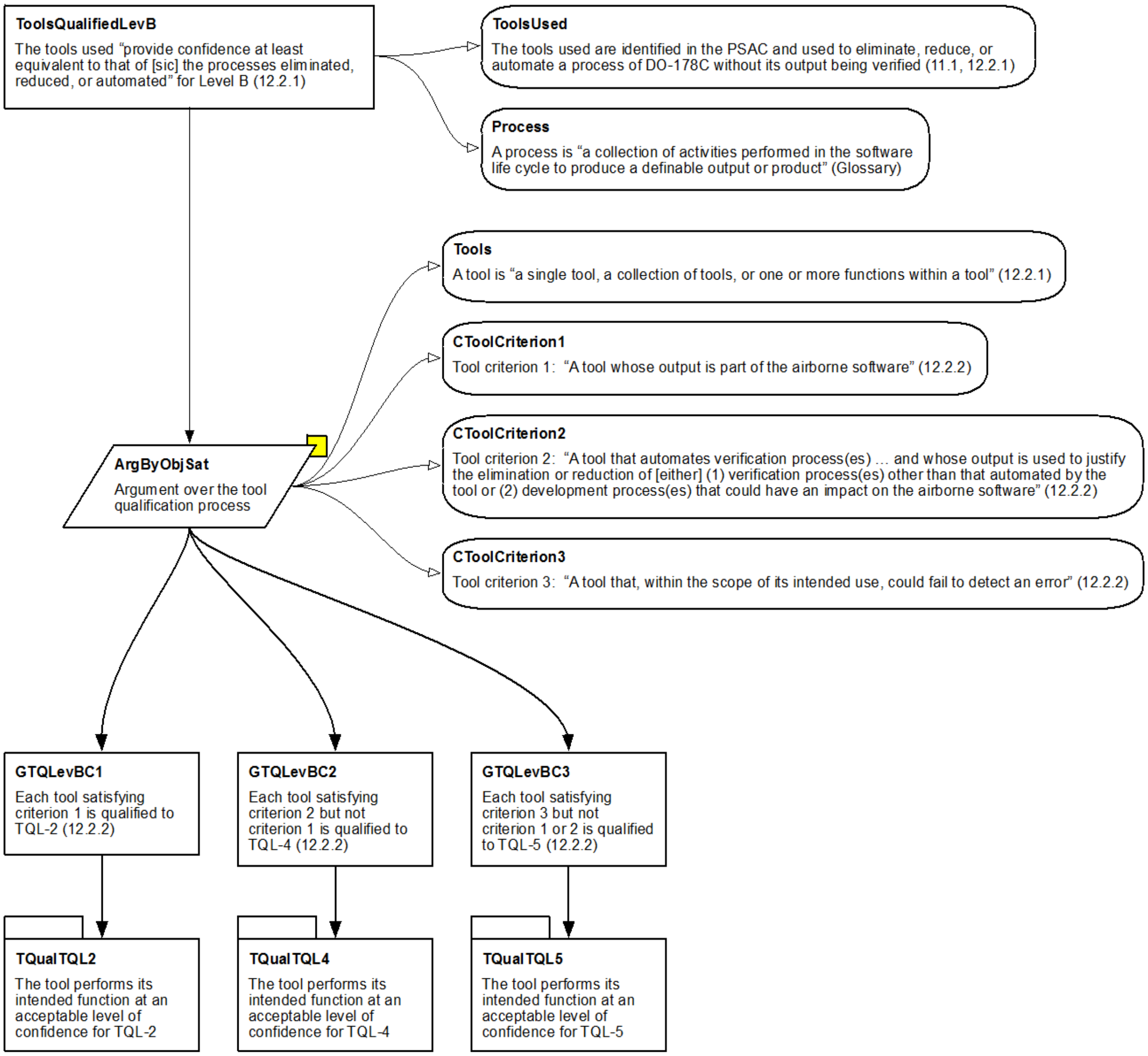

ToolsQualifiedLevB
The tools used “provide confidence at least equivalent to that of [sic] the processes eliminated, reduced, or automated” for Level B (12.2.1)
ToolsUsed
The tools used are identified in the PSAC and used to eliminate, reduce, or automate a process of DO-178C without its output being verified (11.1, 12.2.1)
Process
A process is “a collection of activities performed in the software life cycle to produce a definable output or product” (Glossary)
Tools
A tool is “a single tool, a collection of tools, or one or more functions within a tool” (12.2.1)
CToolCriterion1
Tool criterion 1: “A tool whose output is part of the airborne software” (12.2.2)
CToolCriterion2
Tool criterion 2: “A tool that automates verification process(es) … and whose output is used to justify the elimination or reduction of [either] (1) verification process(es) other than that automated by the tool or (2) development process(es) that could have an impact on the airborne software” (12.2.2)
CToolCriterion3
Tool criterion 3: “A tool that, within the scope of its intended use, could fail to detect an error” (12.2.2)
ArgByObjSat
Argument over the tool qualification process
GTQLevBC1
Each tool satisfying criterion 1 is qualified to TQL-2 (12.2.2)
GTQLevBC2
Each tool satisfying criterion 2 but not criterion 1 is qualified to TQL-4 (12.2.2)
GTQLevBC3
Each tool satisfying criterion 3 but not criterion 1 or 2 is qualified to TQL-5 (12.2.2)
TQualTQL2
The tool performs its intended function at an acceptable level of confidence for TQL-2
TQualTQL4
The tool performs its intended function at an acceptable level of confidence for TQL-4
TQualTQL5
The tool performs its intended function at an acceptable level of confidence for TQL-5

**Figure B-5. ToolsQualifedLevB**

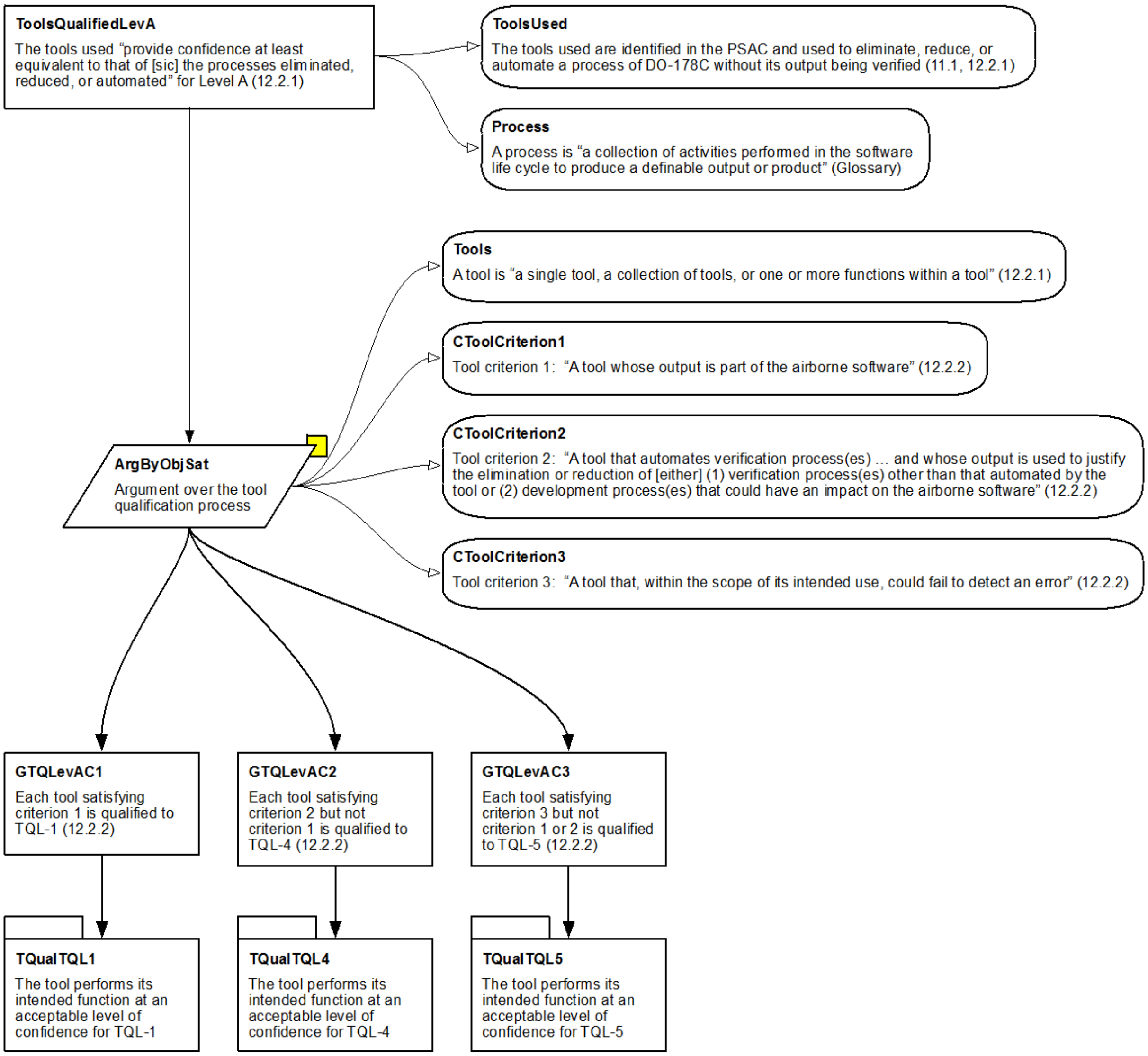

ToolsQualifiedLevA
The tools used "provide confidence at least equivalent to that of [sic] the processes eliminated, reduced, or automated" for Level A (12.2.1)
ToolsUsed
The tools used are identified in the PSAC and used to eliminate, reduce, or automate a process of DO-178C without its output being verified (11.1, 12.2.1)
Process
A process is "a collection of activities performed in the software life cycle to produce a definable output or product" (Glossary)
Tools
A tool is "a single tool, a collection of tools, or one or more functions within a tool" (12.2.1)
CToolCriterion1
Tool criterion 1: "A tool whose output is part of the airborne software" (12.2.2)
CToolCriterion2
Tool criterion 2: "A tool that automates verification process(es) … and whose output is used to justify the elimination or reduction of [either] (1) verification process(es) other than that automated by the tool or (2) development process(es) that could have an impact on the airborne software" (12.2.2)
CToolCriterion3
Tool criterion 3: "A tool that, within the scope of its intended use, could fail to detect an error" (12.2.2)
ArgByObjSat
Argument over the tool qualification process
GTQLevAC1
Each tool satisfying criterion 1 is qualified to TQL-1 (12.2.2)
GTQLevAC2
Each tool satisfying criterion 2 but not criterion 1 is qualified to TQL-4 (12.2.2)
GTQLevAC3
Each tool satisfying criterion 3 but not criterion 1 or 2 is qualified to TQL-5 (12.2.2)
TQualTQL1
The tool performs its intended function at an acceptable level of confidence for TQL-1
TQualTQL4
The tool performs its intended function at an acceptable level of confidence for TQL-4
TQualTQL5
The tool performs its intended function at an acceptable level of confidence for TQL-5

**Figure B-6. ToolsQualifedLevA**

## B.3. TOOL QUALIFICATION ARGUMENT FOR TQL-5

Figure B-7 depicts the argument supporting the conclusion that a tool "performs its intended function at an acceptable level of confidence for TQL-5." This argument is similar to the DO-178C arguments in support of the conclusion that airborne software performs its acceptable level of safety at the appropriate software level.

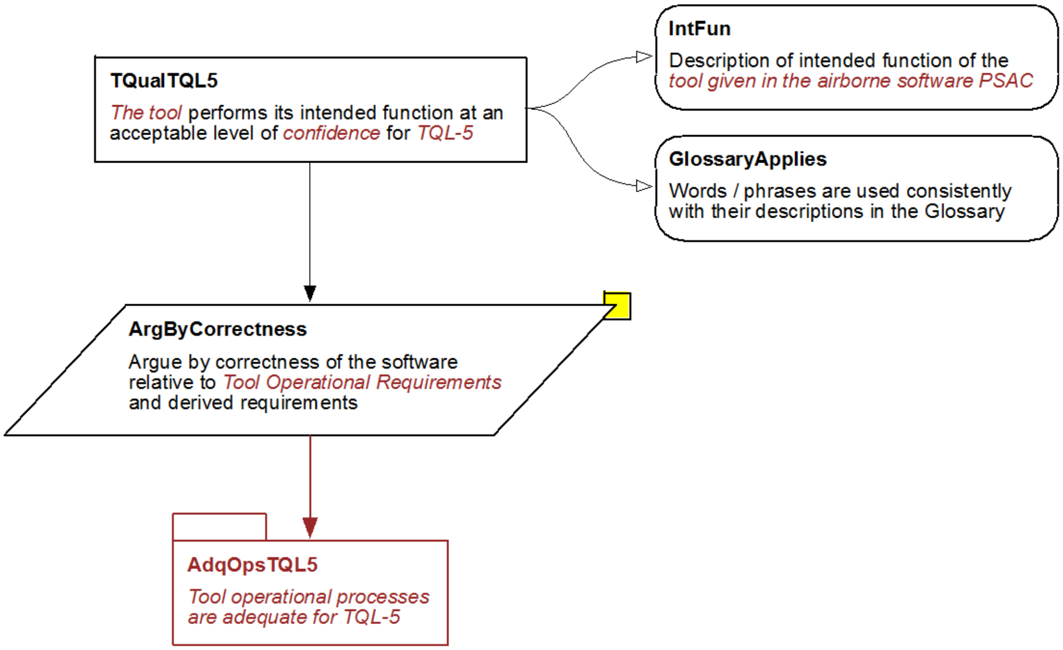

TQualTQL5
The tool performs its intended function at an acceptable level of confidence for TQL-5
IntFun
Description of intended function of the tool given in the airborne software PSAC
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
ArgByCorrectness
Argue by correctness of the software relative to Tool Operational Requirements and derived requirements
AdqOpsTQL5
Tool operational processes are adequate for TQL-5

**Figure B-7. TQUALTQL5**

Red color and italic type in this and subsequent figures indicate differences from the analogous DO-178C arguments. In this cases, the main differences are:

- The argument applies to a tool at a TQL, not to the airborne software at a level.
- The tool's intended function is documented in the airborne software's PSAC rather than in the tool's own documentation.
- The tool is argued to be correct with regard to its tool operational requirements rather than to allocated system requirements.
- Correctness is inferred from the adequacy of the tool operational processes (which is supported by tool-level verification and validation results among other evidence).

Section B.3.1 gives the argument for the adequacy of the tool operational process. Section B.3.2 gives the argument for confidence in the TQL-5 tool-qualification argument.

### B.3.1. EVIDENCE OF CORRECTNESS FOR TQL-5

Figure B-7 gives the argument that the tool performs its intended function at an acceptable level of confidence for TQL-5. This functionality is inferred from the tool’s correctness with respect to its tool operational requirements. That correctness is in turn inferred from the adequacy of the tool operational processes. Figure B-8 presents the argument supporting that premise, which argues over satisfaction of tool operational process objectives that apply at TQL-5. Premises representing each objective show that:

- The tool is identified, its intended use described, the need for tool qualification defined, the TQL determined, the stakeholders identified, and the tool operational requirements described (TQNeedEst).
- The tool operational requirements are defined in sufficient detail (TORsDefined) and are correct (TORsCorrect).
- The tool has been installed correctly in the airborne software development environment (TEOCInstd).
- The functionality and outputs of the installed tool are verified to comply with the tool operational requirements (TOpComplies), and the tool’s satisfaction of the airborne software-development process needs is validated (SWLCPNMeet)

That is, the tool operational process is adequate because the highest-level tool requirements are adequate and the tool has been installed correctly, verified, and validated in its airborne software-development context.

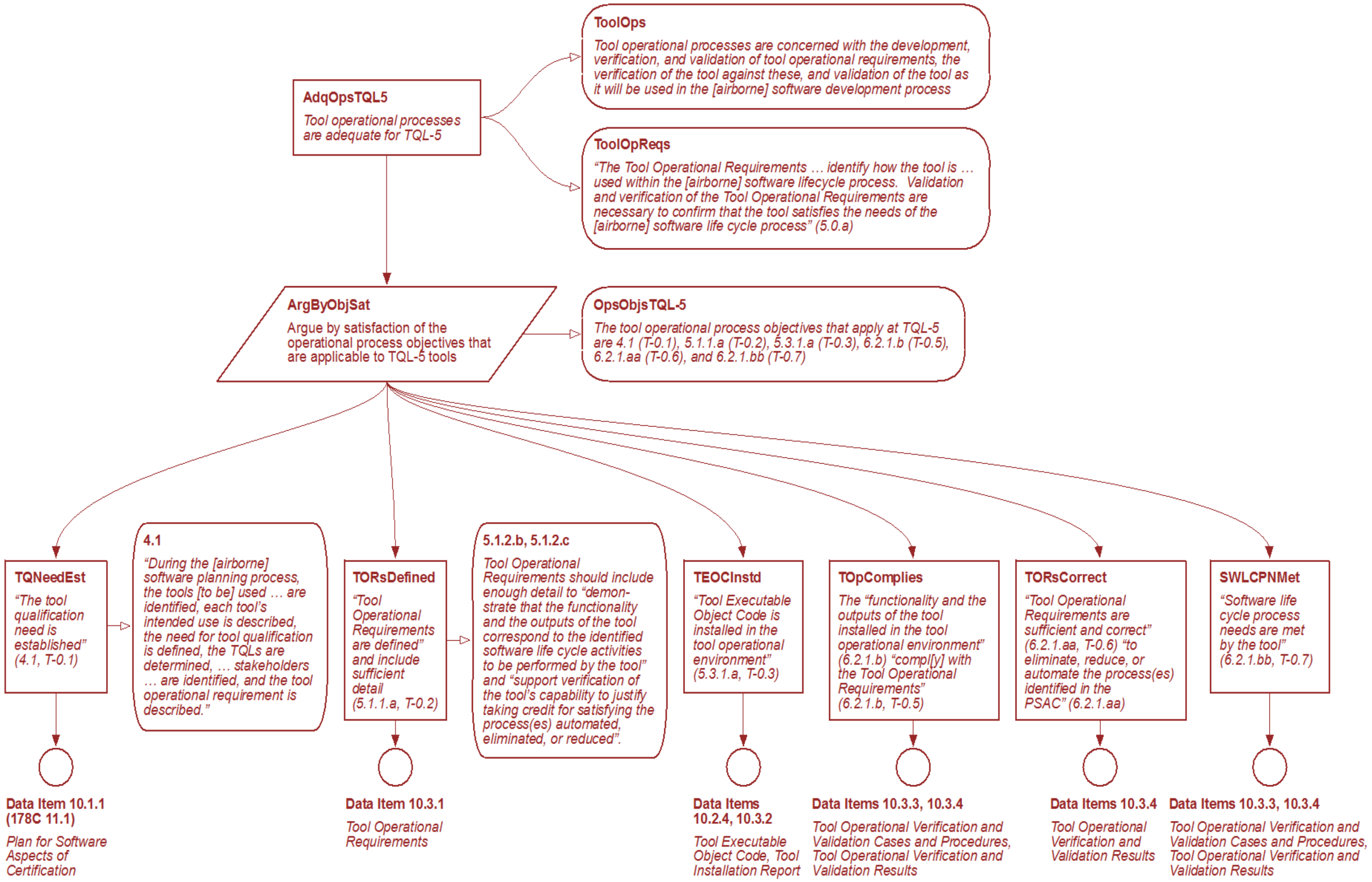

AdqOpsTQL5
Tool operational processes are adequate for TQL-5
ToolOps
Tool operational processes are concerned with the development, verification, and validation of tool operational requirements, the verification of the tool against these, and validation of the tool as it will be used in the [airborne] software development process
ToolOpReqs
“The Tool Operational Requirements … identify how the tool is … used within the [airborne] software lifecycle process. Validation and verification of the Tool Operational Requirements are necessary to confirm that the tool satisfies the needs of the [airborne] software life cycle process” (5.0.a)
ArgByObjSat
Argue by satisfaction of the operational process objectives that are applicable to TQL-5 tools
OpsObjsTQL-5
The tool operational process objectives that apply at TQL-5 are 4.1 (T-0.1), 5.1.1.a (T-0.2), 5.3.1.a (T-0.3), 6.2.1.b (T-0.5), 6.2.1.aa (T-0.6), and 6.2.1.bb (T-0.7)
TQNeedEst
“The tool qualification need is established” (4.1, T-0.1)
4.1
“During the [airborne] software planning process, the tools [to be] used … are identified, each tool’s intended use is described, the need for tool qualification is defined, the TQLs are determined, … stakeholders … are identified, and the tool operational requirement is described.”
TORsDefined
“Tool Operational Requirements are defined” and include sufficient detail (5.1.1.a, T-0.2)
5.1.2.b, 5.1.2.c
Tool Operational Requirements should include enough detail to “demonstrate that the functionality and the outputs of the tool correspond to the identified software life cycle activities to be performed by the tool” and “support verification of the tool’s capability to justify taking credit for satisfying the process(es) automated, eliminated, or reduced”.
TEOCInstd
“Tool Executable Object Code is installed in the tool operational environment” (5.3.1.a, T-0.3)
TOpComplies
The “functionality and the outputs of the tool installed in the tool operational environment” (6.2.1.b) “compl[y] with the Tool Operational Requirements” (6.2.1.b, T-0.5)
TORsCorrect
“Tool Operational Requirements are sufficient and correct” (6.2.1.aa, T-0.6) “to eliminate, reduce, or automate the process(es) identified in the PSAC” (6.2.1.aa)
SWLCPNMet
“Software life cycle process needs are met by the tool” (6.2.1.bb, T-0.7)
Data Item 10.1.1 (178C 11.1)
Plan for Software Aspects of Certification
Data Item 10.3.1
Tool Operational Requirements
Data Items 10.2.4, 10.3.2
Tool Executable Object Code, Tool Installation Report
Data Items 10.3.3, 10.3.4
Tool Operational Verification and Validation Cases and Procedures, Tool Operational Verification and Validation Results
Data Items 10.3.4
Tool Operational Verification and Validation Results
Data Items 10.3.3, 10.3.4
Tool Operational Verification and Validation Cases and Procedures, Tool Operational Verification and Validation Results

**Figure B-8. AdqOpsTQL5B.3.2. Confidence in Tool-Qualification Argument for TQL-5**

As in the DO-178C argument, confidence in demonstrating that software is correct with respect to its requirements is sufficient to show that the software will perform its intended function would be undermined by the use of inadequate processes. Figure B-9 depicts the confidence argument for TQL-5. The argument infers that "the evidence provided is adequate for justifying that the correctness of the tool has been demonstrated to the extent needed for TQL-5" on the grounds that adequate configuration management has been conducted, adequate tool quality-assurance is in place, and the certification liaison process is adequate.

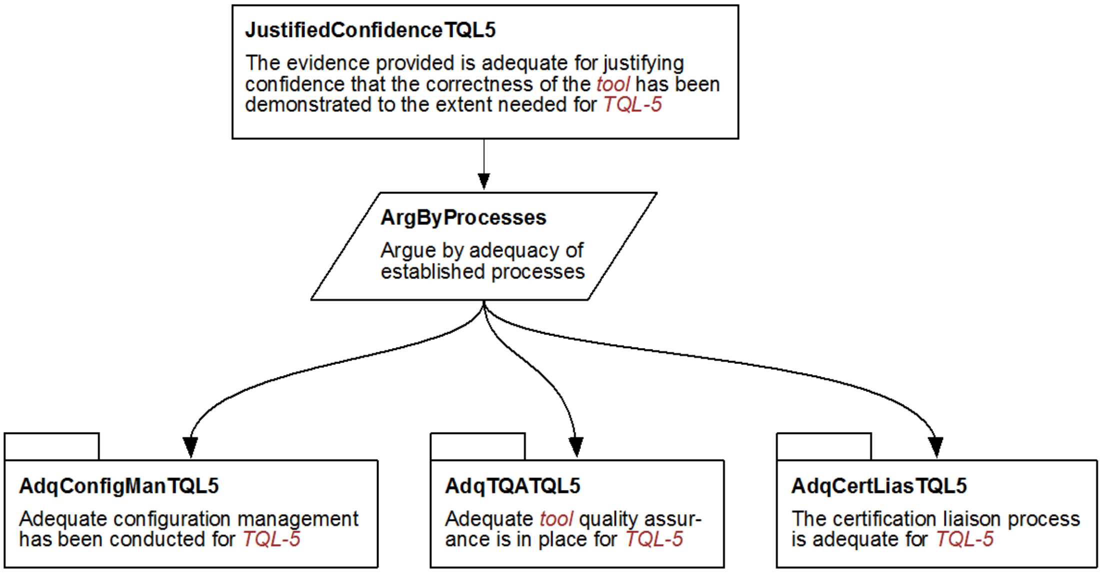

JustifiedConfidenceTQL5
The evidence provided is adequate for justifying confidence that the correctness of the tool has been demonstrated to the extent needed for TQL-5
ArgByProcesses
Argue by adequacy of established processes
AdqConfigManTQL5
Adequate configuration management has been conducted for TQL-5
AdqTQATQL5
Adequate tool quality assurance is in place for TQL-5
AdqCertLiasTQL5
The certification liaison process is adequate for TQL-5

**Figure B-9. JustifiedConfidenceTQL5**

Figure B-10 depicts the argument that configuration management is adequate for a TQL-5 tool. This argument—largely analogous to the corresponding DO-178C argument (namely AdqConfigManLevD)—infers the adequacy of configuration management from satisfaction of the applicable objectives. Premises representing these objectives cite evidence to show that:

- The versions of configuration items, such as requirements and the source code, are identified unambiguously (ConfItemsLabeled).
- Configuration allows identification of the development artifacts associated with any tool release and ensures that only authorized tool releases are used to develop the airborne software (ArcRelEst).

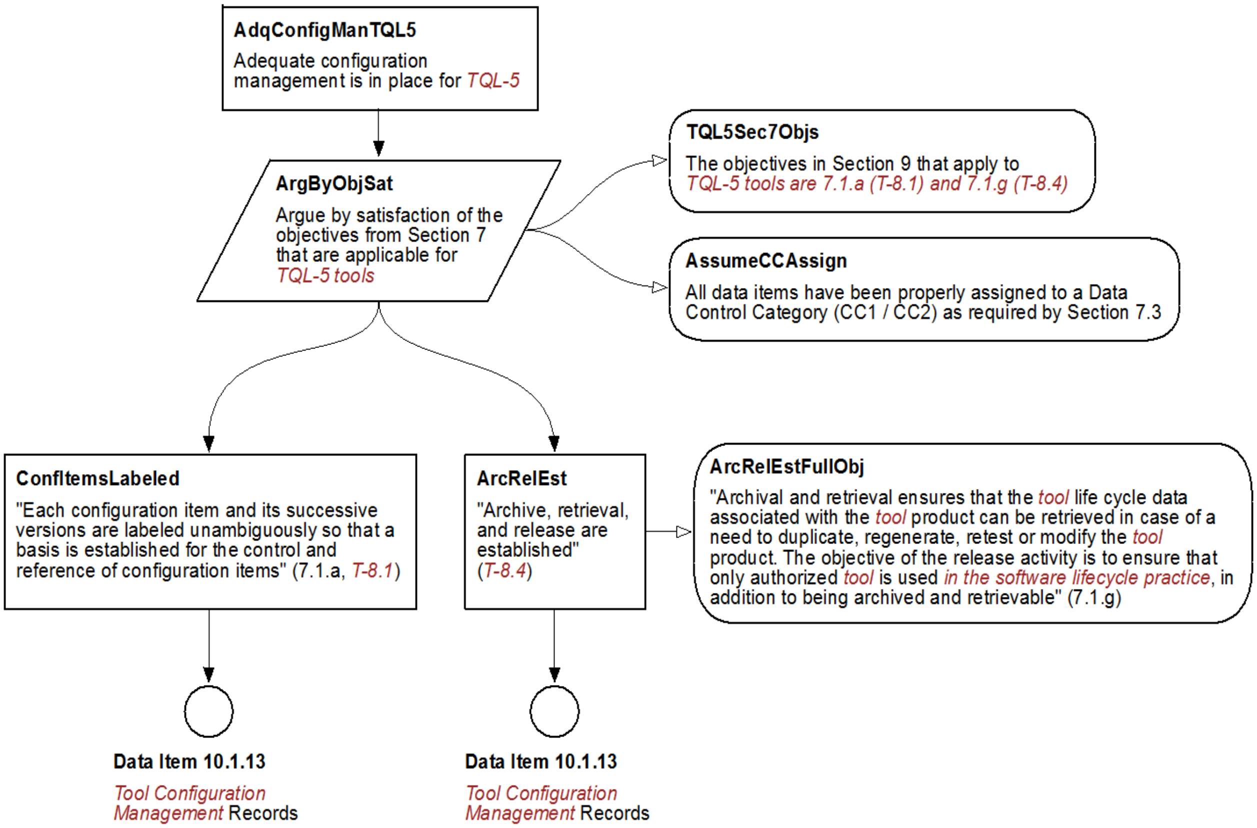

AdqConfigManTQL5
Adequate configuration management is in place for TQL-5
ArgByObjSat
Argue by satisfaction of the objectives from Section 7 that are applicable for TQL-5 tools
TQL5Sec7Objs
The objectives in Section 9 that apply to TQL-5 tools are 7.1.a (T-8.1) and 7.1.g (T-8.4)
AssumeCCAssign
All data items have been properly assigned to a Data Control Category (CC1 / CC2) as required by Section 7.3
ConfItemsLabeled
"Each configuration item and its successive versions are labeled unambiguously so that a basis is established for the control and reference of configuration items" (7.1.a, T-8.1)
ArcRelEst
"Archive, retrieval, and release are established" (T-8.4)
ArcRelEstFullObj
"Archival and retrieval ensures that the tool life cycle data associated with the tool product can be retrieved in case of a need to duplicate, regenerate, retest or modify the tool product. The objective of the release activity is to ensure that only authorized tool is used in the software lifecycle practice, in addition to being archived and retrievable" (7.1.g)
Data Item 10.1.13
Tool Configuration Management Records
Data Item 10.1.13
Tool Configuration Management Records

**Figure B-10. AdqConfigManTQL5**

Figure B-11 depicts the argument that tool quality-assurance processes are adequate for a TQL-5 tool. This argument—also largely analogous to the corresponding DO-178C argument (namely AdqSQALevD)—infers the adequacy of tool quality assurance from satisfaction of the applicable objectives. Premises representing these objectives cite evidence from an independent review of processes compliance with approved plans (AssureCompPlans) and an independent conformity review (AssureConfRef).

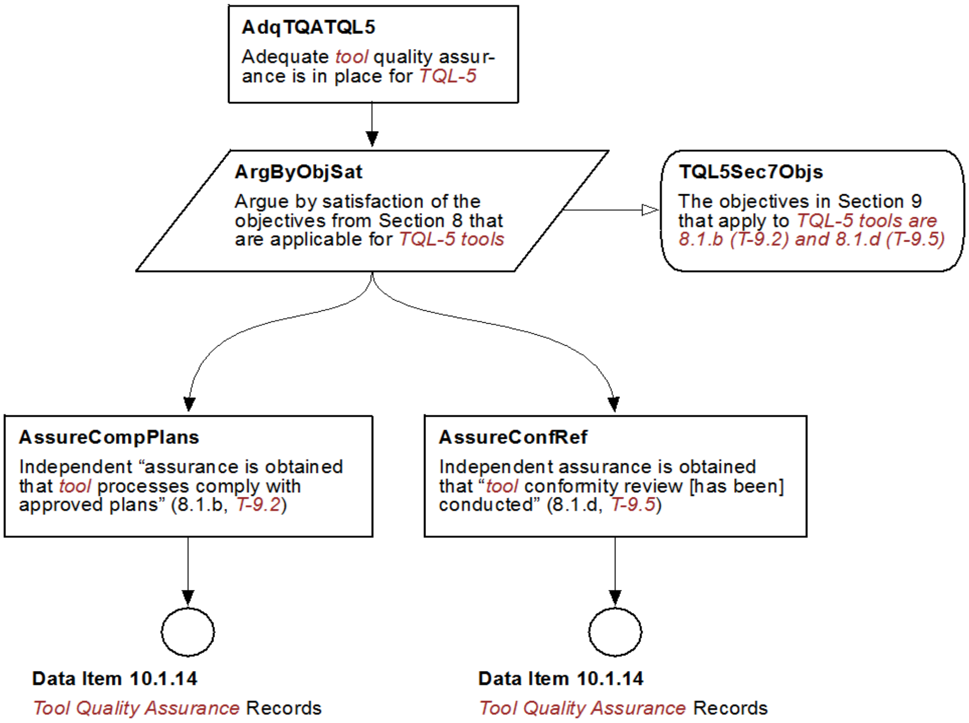

AdqTQATQL5
Adequate *tool* quality assurance is in place for *TQL-5*
ArgByObjSat
Argue by satisfaction of the objectives from Section 8 that are applicable for *TQL-5 tools*
TQL5Sec7Objs
The objectives in Section 9 that apply to *TQL-5 tools are 8.1.b (T-9.2) and 8.1.d (T-9.5)*
AssureCompPlans
Independent "assurance is obtained that *tool* processes comply with approved plans" (8.1.b, *T-9.2*)
AssureConfRef
Independent assurance is obtained that "*tool* conformity review [has been] conducted" (8.1.d, *T-9.5*)
Data Item 10.1.14
*Tool Quality Assurance* Records
Data Item 10.1.14
*Tool Quality Assurance* Records

**Figure B-11. AdqTQATQL5**

Figure B-12 depicts the argument that the certification liaison process is adequate for a TQL-5 tool. This argument—somewhat analogous to the corresponding DO-178C argument (namely AdqCertLiasLevD)—infers the adequacy of the liaison process from satisfaction of the applicable objectives. Premises representing these objectives cite evidence to show that:

- There is communication and understanding between the applicant and the certification authority (CertAutComm).
- The means of compliance is proposed and agreed (ComplianceAgree).
- Evidence of compliance is provided (ComplianceSubs).
- Known problems are examined to determine whether they undermine satisfaction of the tool operational requirements (ImpactAssessed).

Section B.1. discusses key differences between the DO-330 and DO-178C certification contexts. Those differences are reflected in differences between this argument and its DO-178C analogue:

- Documentation of the tool's means of compliance to DO-330 is contained in part in the Plan for Software Aspects of Certification of the airborne software it is used to develop.
- Analysis of the impact of reported problems is explicitly a DO-330 process.

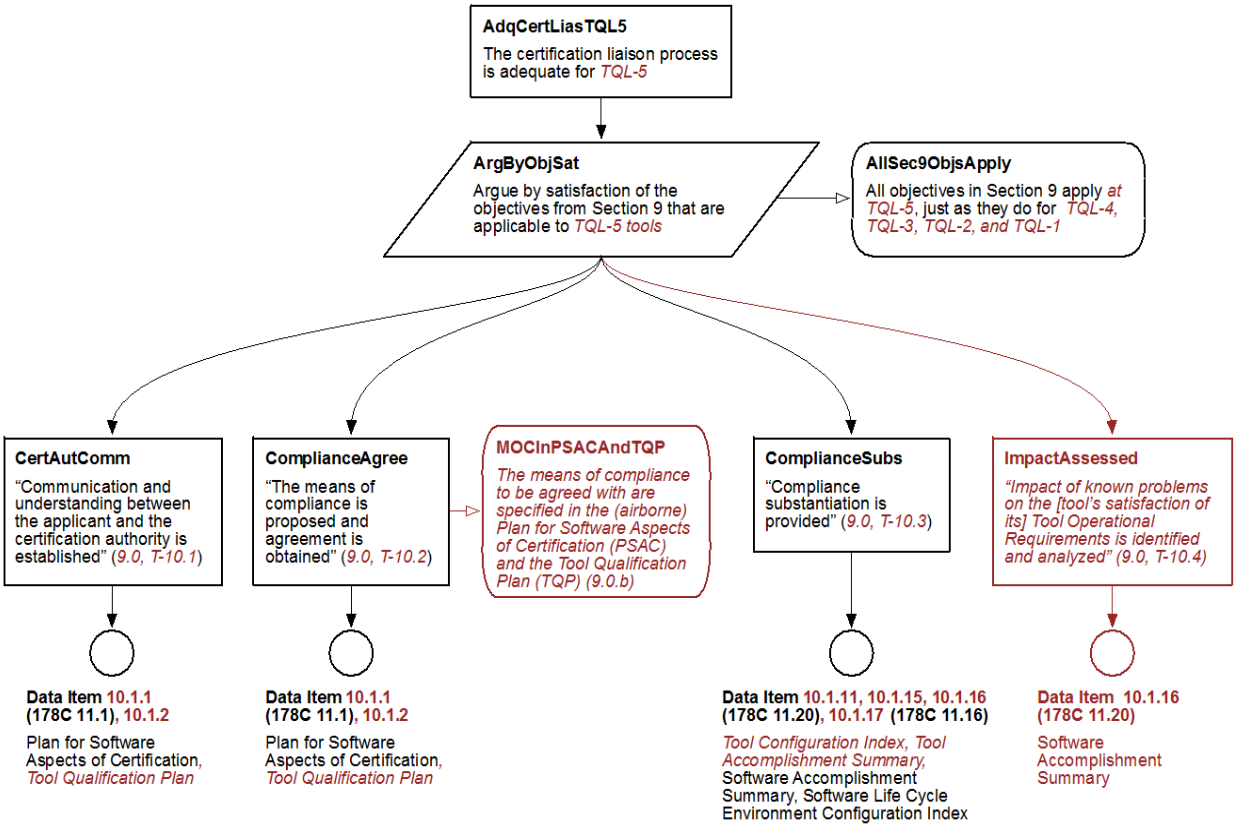

AdqCertLiasTQL5
The certification liaison process is adequate for TQL-5
ArgByObjSat
Argue by satisfaction of the objectives from Section 9 that are applicable to TQL-5 tools
AllSec9ObjsApply
All objectives in Section 9 apply at TQL-5, just as they do for TQL-4, TQL-3, TQL-2, and TQL-1
CertAutComm
"Communication and understanding between the applicant and the certification authority is established" (9.0, T-10.1)
ComplianceAgree
"The means of compliance is proposed and agreement is obtained" (9.0, T-10.2)
MOCInPSACAndTQP
The means of compliance to be agreed with are specified in the (airborne) Plan for Software Aspects of Certification (PSAC) and the Tool Qualification Plan (TQP) (9.0.b)
ComplianceSubs
"Compliance substantiation is provided" (9.0, T-10.3)
ImpactAssessed
"Impact of known problems on the [tool's satisfaction of its] Tool Operational Requirements is identified and analyzed" (9.0, T-10.4)
Data Item 10.1.1 (178C 11.1), 10.1.2
Plan for Software Aspects of Certification, Tool Qualification Plan
Data Item 10.1.1 (178C 11.1), 10.1.2
Plan for Software Aspects of Certification, Tool Qualification Plan
Data Item 10.1.11, 10.1.15, 10.1.16 (178C 11.20), 10.1.17 (178C 11.16)
Tool Configuration Index, Tool Accomplishment Summary, Software Accomplishment Summary, Software Life Cycle Environment Configuration Index
Data Item 10.1.16 (178C 11.20)
Software Accomplishment Summary

**Figure B-12. AdqCertLiasTQL5**

### B.3.2 CONFIDENCE IN TOOL QUALIFICATION ARGUMENT FOR TQL-5

As in the DO-178C argument, confidence that demonstrating that software is correct with respect to its requirements is sufficient to show that the software will perform its intended function would be undermined by the use of inadequate processes. Figure B-13 depicts the confidence argument for TQL-5. The argument infers that “the evidence provided is adequate for justifying that the correctness of the tool has been demonstrated to the extent needed for TQL-5” on the grounds that adequate configuration management has been conducted, adequate tool quality assurance is in place, and the certification liaison process is adequate.

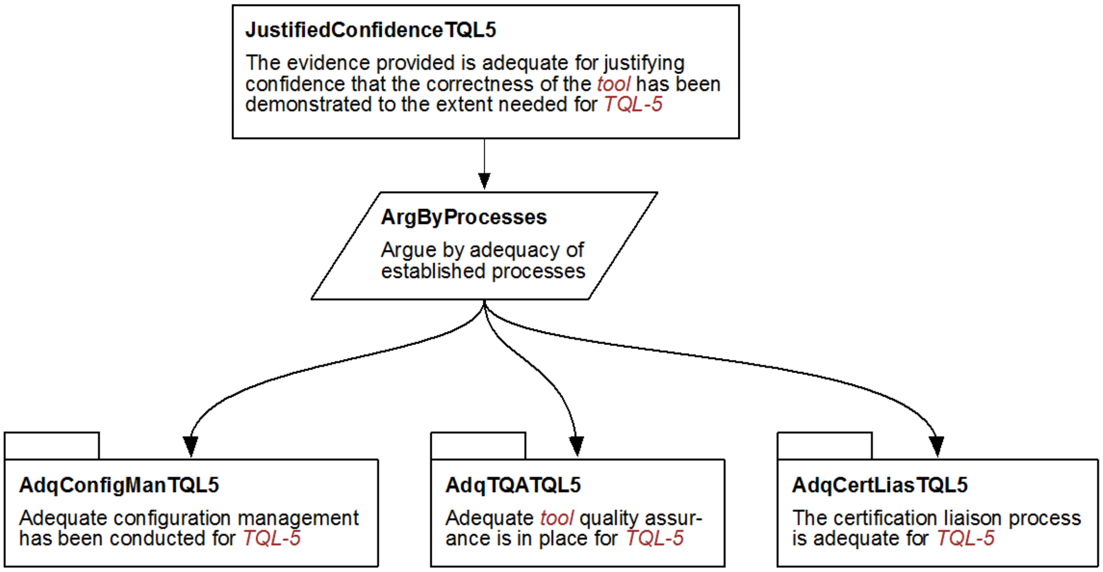

JustifiedConfidenceTQL5
The evidence provided is adequate for justifying confidence that the correctness of the tool has been demonstrated to the extent needed for TQL-5
ArgByProcesses
Argue by adequacy of established processes
AdqConfigManTQL5
Adequate configuration management has been conducted for TQL-5
AdqTQATQL5
Adequate tool quality assurance is in place for TQL-5
AdqCertLiasTQL5
The certification liaison process is adequate for TQL-5

**Figure B-13. JustifiedConfidenceTQL5**

## B.4. TOOL QUALIFICATION ARGUMENT FOR TQL-4

Figure B-14 depicts the argument supporting the conclusion that a tool “performs its intended function at an acceptable level of confidence for TQL-4.” This argument is similar to the tool qualification argument for TQL-5 and the DO-178C arguments in support of the conclusion that airborne software performs its acceptable level of safety at the appropriate software level.

This argument revises the TQL-5 argument’s premise about tool operational process adequacy to reflect the higher tool qualification level. It also adds two new premises, namely that the tool requirements are a satisfactory refinement of the tool operational requirements and the tool executable object code satisfactorily refines the tool requirements. The addition of a layer of more detailed requirements and evidence of their satisfaction at a higher tool qualification level reflects similar additions in the DO-178C argument as the software level increases.

Section B.4.1 presents the argument for the adequacy of the tool operational process. Section B.4.2 presents the argument for confidence in the TQL-4 tool-qualification argument.

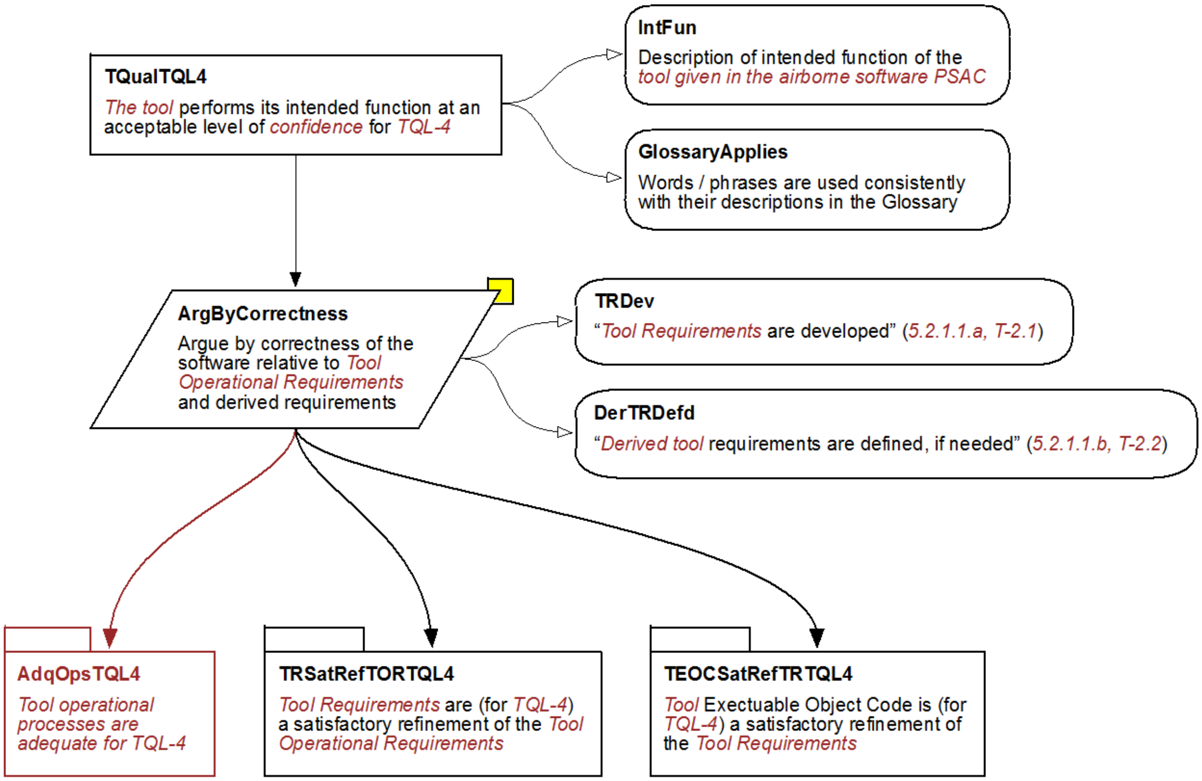

TQualTQL4
The tool performs its intended function at an acceptable level of confidence for TQL-4
IntFun
Description of intended function of the tool given in the airborne software PSAC
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
ArgByCorrectness
Argue by correctness of the software relative to Tool Operational Requirements and derived requirements
TRDev
"Tool Requirements are developed" (5.2.1.1.a, T-2.1)
DerTRDefd
"Derived tool requirements are defined, if needed" (5.2.1.1.b, T-2.2)
AdqOpsTQL4
Tool operational processes are adequate for TQL-4
TRSatRefTORTQL4
Tool Requirements are (for TQL-4) a satisfactory refinement of the Tool Operational Requirements
TEOCSatRefTRTQL4
Tool Exectuable Object Code is (for TQL-4) a satisfactory refinement of the Tool Requirements

**Figure B-14. TQualTQL4**

### B.4.1. EVIDENCE OF CORRECTNESS FOR TQL-4

Figure B-14 presents the argument that the tool performs its intended function at an acceptable level of confidence for TQL-4. This functionality is inferred from the tool's correctness with respect to its tool operational requirements, which is in turn inferred from:

- The adequacy of the tool operational processes
- The tool requirements adequate refining the tool operational requirements
- The tool satisfying its tool requirements

Figure B-15 presents the argument supporting the first of those premises. This argument extends the corresponding TQL-5 argument (see figure B-8) with one additional evidence-supported premise, namely that the "Tool Operational Requirements are complete, accurate, verifiable, and consistent" (TORsAdequate).

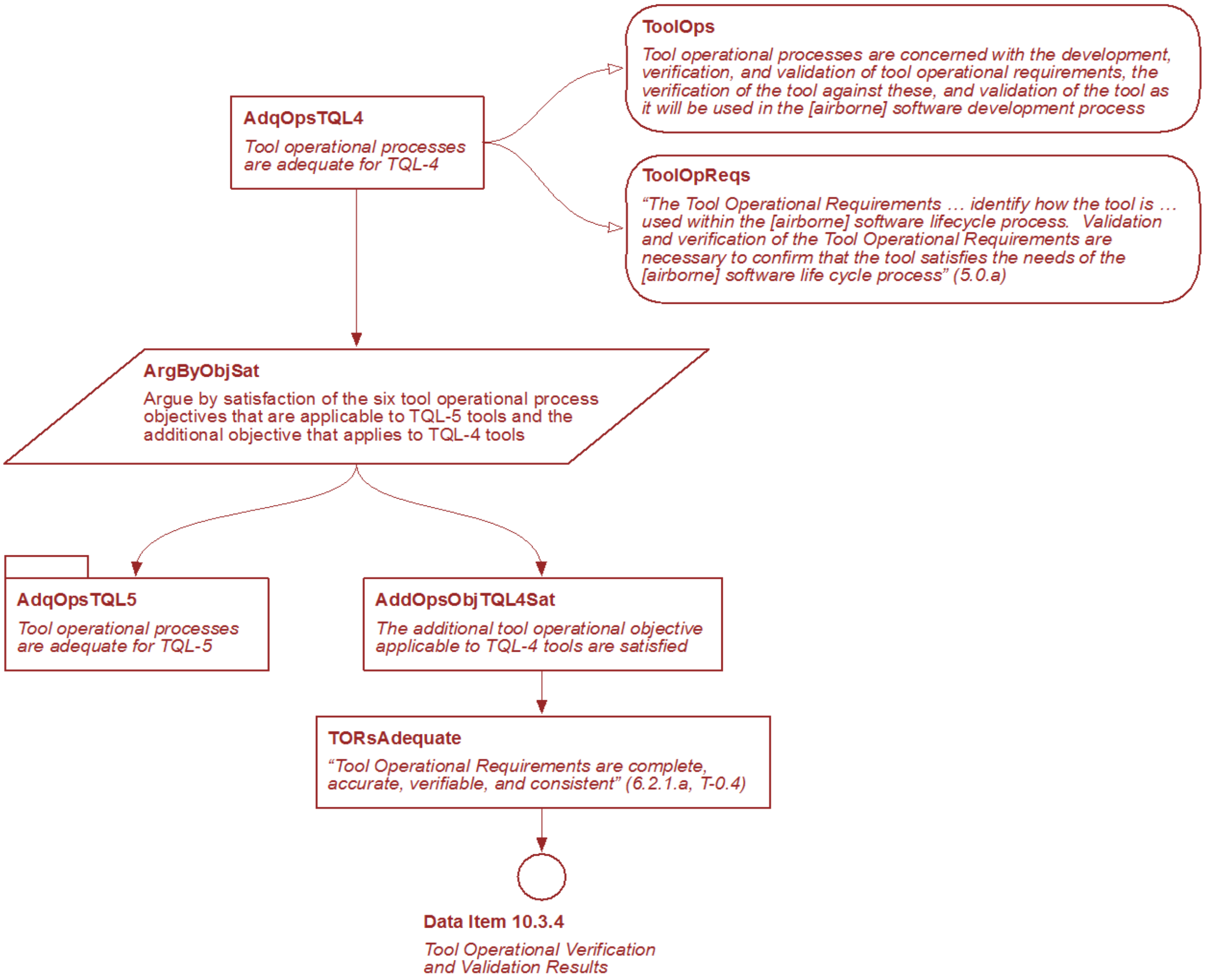

ToolOps
Tool operational processes are concerned with the development, verification, and validation of tool operational requirements, the verification of the tool against these, and validation of the tool as it will be used in the [airborne] software development process
AdqOpsTQL4
Tool operational processes are adequate for TQL-4
ToolOpReqs
"The Tool Operational Requirements … identify how the tool is … used within the [airborne] software lifecycle process. Validation and verification of the Tool Operational Requirements are necessary to confirm that the tool satisfies the needs of the [airborne] software life cycle process" (5.0.a)
ArgByObjSat
Argue by satisfaction of the six tool operational process objectives that are applicable to TQL-5 tools and the additional objective that applies to TQL-4 tools
AdqOpsTQL5
Tool operational processes are adequate for TQL-5
AddOpsObjTQL4Sat
The additional tool operational objective applicable to TQL-4 tools are satisfied
TORsAdequate
"Tool Operational Requirements are complete, accurate, verifiable, and consistent" (6.2.1.a, T-0.4)
Data Item 10.3.4
Tool Operational Verification and Validation Results

**Figure B-15. AdqOpsTQL4**

Figure B-16 presents the argument supporting the premise that the “Tool Requirements are (for TQL-4) a satisfactory refinement of the Tool Operational Requirements.” The support for this conclusion comes from satisfaction of the applicable objectives and is largely analogous to the corresponding parts of the DO-178C argument (HLRSatSRRefLevC and HLRSatSRRefLevD):

- The compliance (TRComply), consistency (TRAccCons), and traceability (TRAccCons) premises have analogues in the DO-178C Level D argument.
- The verifiability (TRVerifiable) and algorithm accuracy (AlgorAcc) premises have analogues in the DO-178C Level C argument.
- The requirements compatibility (TROECompatDef), error conditions (TRErrCondDef), and user instructions (TRUIDef) premises are tool-related analogues to the compatibility with target computer premises that apply to the (embedded) airborne software at Level B.
- TQL-4 does not have an analogue to the DO-178C Level B argument’s premise that high-level airborne software requirements conform to the Software Requirements Standards. (That requirement is introduced at TQL-3.)

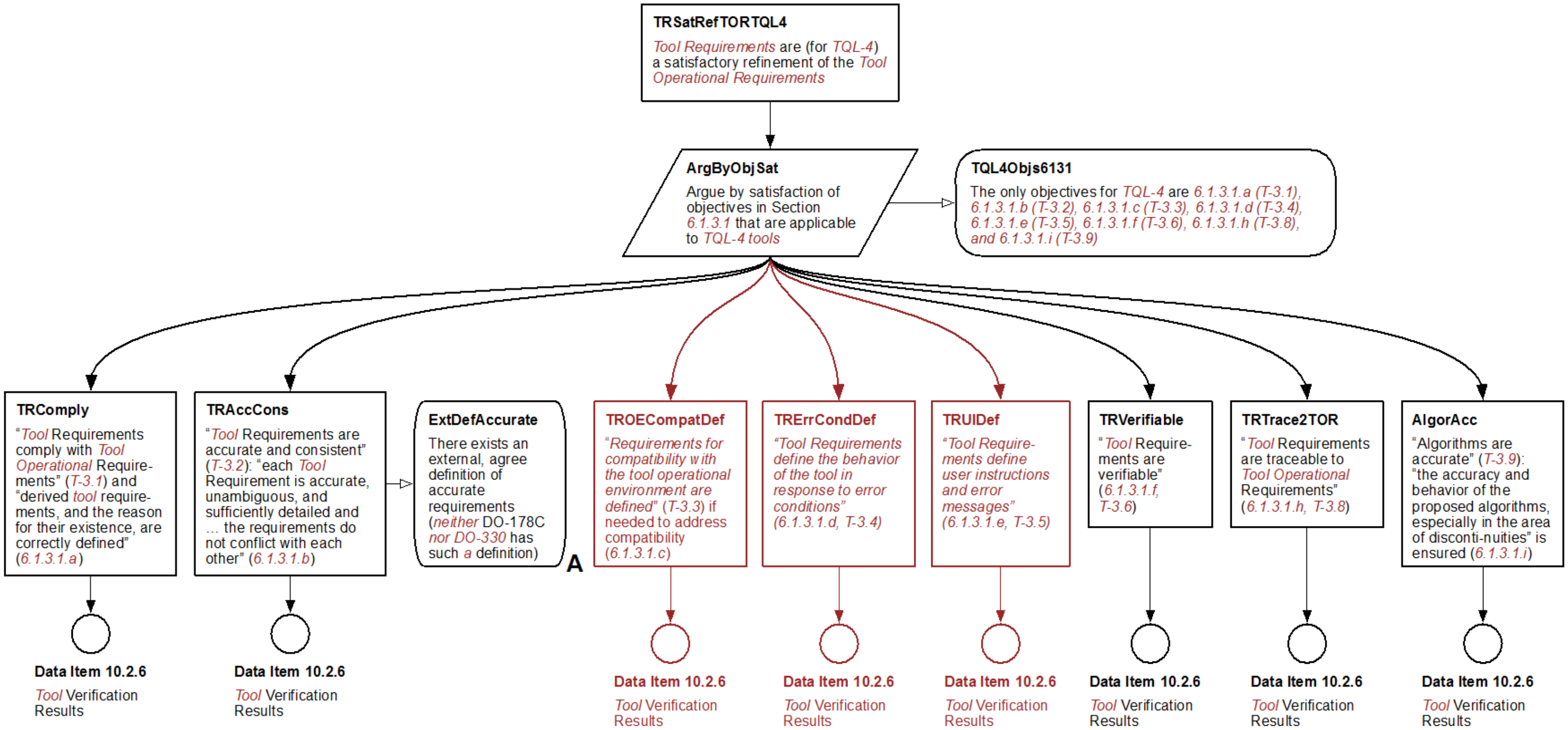

TRSatRefTORTQL4
Tool Requirements are (for TQL-4) a satisfactory refinement of the Tool Operational Requirements
ArgByObjSat
Argue by satisfaction of objectives in Section 6.1.3.1 that are applicable to TQL-4 tools
TQL4Objs6131
The only objectives for TQL-4 are 6.1.3.1.a (T-3.1), 6.1.3.1.b (T-3.2), 6.1.3.1.c (T-3.3), 6.1.3.1.d (T-3.4), 6.1.3.1.e (T-3.5), 6.1.3.1.f (T-3.6), 6.1.3.1.h (T-3.8), and 6.1.3.1.i (T-3.9)
TRComply
"Tool Requirements comply with Tool Operational Requirements" (T-3.1) and "derived tool requirements, and the reason for their existence, are correctly defined" (6.1.3.1.a)
TRAccCons
"Tool Requirements are accurate and consistent" (T-3.2): "each Tool Requirement is accurate, unambiguous, and sufficiently detailed and … the requirements do not conflict with each other" (6.1.3.1.b)
ExtDefAccurate
There exists an external, agree definition of accurate requirements (neither DO-178C nor DO-330 has such a definition)
A
TROECompatDef
"Requirements for compatibility with the tool operational environment are defined" (T-3.3) if needed to address compatibility (6.1.3.1.c)
TRErrCondDef
"Tool Requirements define the behavior of the tool in response to error conditions" (6.1.3.1.d, T-3.4)
TRUIDef
"Tool Requirements define user instructions and error messages" (6.1.3.1.e, T-3.5)
TRVerifiable
"Tool Requirements are verifiable" (6.1.3.1.f, T-3.6)
TRTrace2TOR
"Tool Requirements are traceable to Tool Operational Requirements" (6.1.3.1.h, T-3.8)
AlgorAcc
"Algorithms are accurate" (T-3.9): "the accuracy and behavior of the proposed algorithms, especially in the area of disconti-nuities" is ensured (6.1.3.1.i)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-16. TRSatRefTORTQL4**

Figure B-17 presents the argument supporting the premise that the "Tool Executable Object Code is (for TQL-4) a satisfactory refinement of the Tool Requirements." This argument is largely analogous to the DO-178C argument that the airborne software's executable object code refines its high-level requirements (EOCSatHLRefLevD). The most prominent differences are:

- Whereas DO-178C is concerned with "software partitioning integrity," DO-330's analogous concern is about "protection mechanisms" (if used), particularly "in multi-function tools" (ProtMechConf).
- DO-178C includes an objective related to the correctness and completeness of parameter data item files (if any) that have no *direct* analogue in DO-330, although the text of the standard makes it clear that any configuration files are to be treated as part of the tool executable object code (see TEOCIsTEOCETC in figure B-16).
- DO-178C includes an objective related to the executable object code's compatibility with the target computer that has no analogue in DO-330.

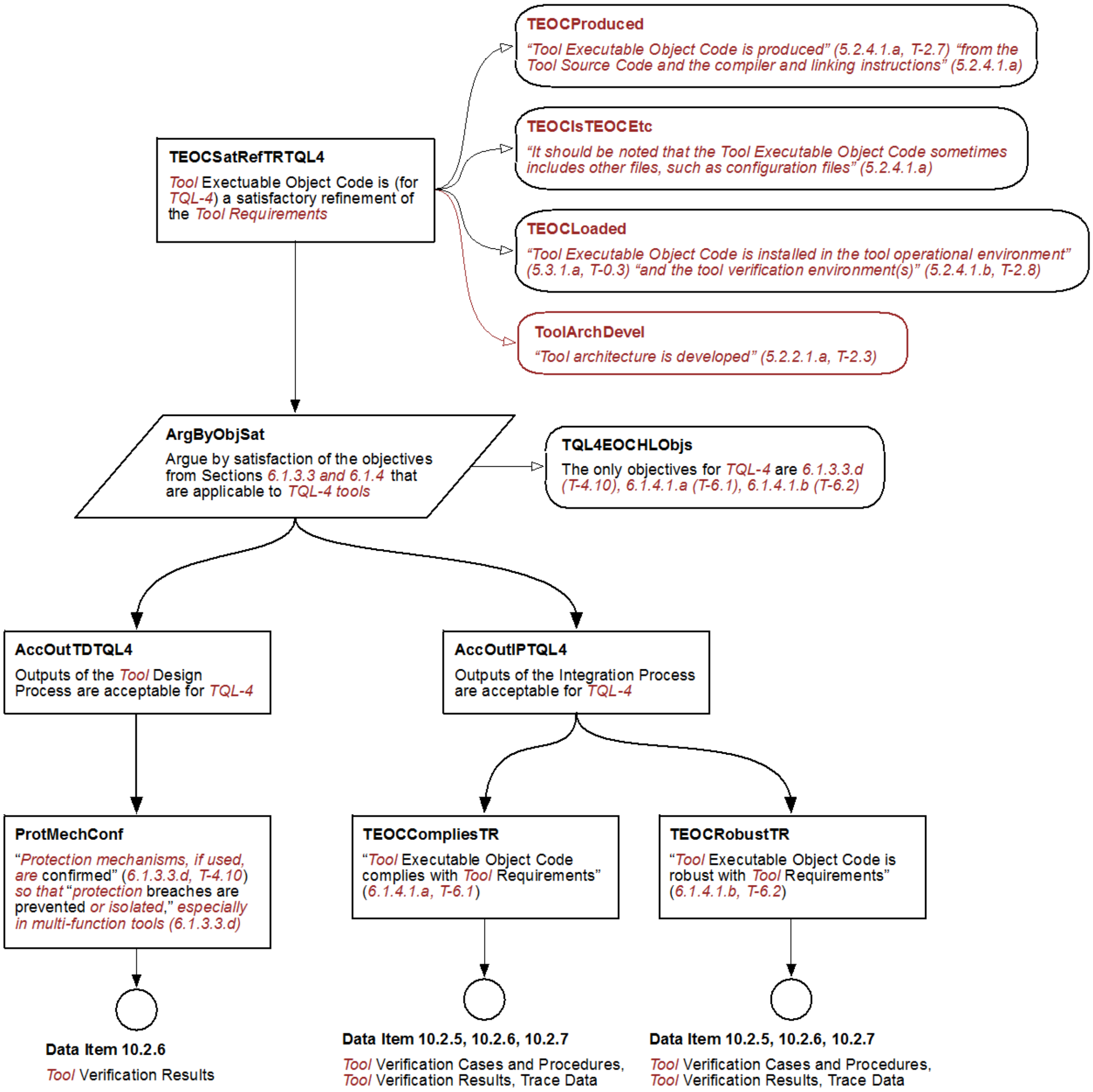

TEOCProduced
"Tool Executable Object Code is produced" (5.2.4.1.a, T-2.7) "from the Tool Source Code and the compiler and linking instructions" (5.2.4.1.a)
TEOCIsTEOCEtc
"It should be noted that the Tool Executable Object Code sometimes includes other files, such as configuration files" (5.2.4.1.a)
TEOCSatRefTRTQL4
Tool Exectuable Object Code is (for TQL-4) a satisfactory refinement of the Tool Requirements
TEOCLoaded
"Tool Executable Object Code is installed in the tool operational environment" (5.3.1.a, T-0.3) "and the tool verification environment(s)" (5.2.4.1.b, T-2.8)
ToolArchDevel
"Tool architecture is developed" (5.2.2.1.a, T-2.3)
ArgByObjSat
Argue by satisfaction of the objectives from Sections 6.1.3.3 and 6.1.4 that are applicable to TQL-4 tools
TQL4EOCHLObjs
The only objectives for TQL-4 are 6.1.3.3.d (T-4.10), 6.1.4.1.a (T-6.1), 6.1.4.1.b (T-6.2)
AccOutTDTQL4
Outputs of the Tool Design Process are acceptable for TQL-4
AccOutIPTQL4
Outputs of the Integration Process are acceptable for TQL-4
ProtMechConf
"Protection mechanisms, if used, are confirmed" (6.1.3.3.d, T-4.10) so that "protection breaches are prevented or isolated," especially in multi-function tools (6.1.3.3.d)
TEOCCompliesTR
"Tool Executable Object Code complies with Tool Requirements" (6.1.4.1.a, T-6.1)
TEOCRobustTR
"Tool Executable Object Code is robust with Tool Requirements" (6.1.4.1.b, T-6.2)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.5, 10.2.6, 10.2.7
Tool Verification Cases and Procedures, Tool Verification Results, Trace Data
Data Item 10.2.5, 10.2.6, 10.2.7
Tool Verification Cases and Procedures, Tool Verification Results, Trace Data

**Figure B-17. TEOCSatRefTRTQL4**

## B.4.2. CONFIDENCE IN TOOL-QUALIFICATION ARGUMENT FOR TQL-4

As in the DO-178C argument, confidence demonstrating that software is correct with respect to its requirements is sufficient to show that the software will perform its intended function would be undermined by the use of inadequate processes. Figure B-18 depicts the confidence argument for TQL-4. The argument infers that "the evidence provided is adequate for justifying that the correctness of the tool has been demonstrated to the extent needed for TQL-4" on the grounds of three premises applied at TQL-5 and two additional premises. The existing premises are:

1. Adequate tool quality assurance is in place (AdqTQATQL5).
2. The certification liaison process is adequate (AdqCertLiasTQL5).
3. Adequate configuration management has been conducted (AdqConfigManTQL4).

The supporting argument for the former two premises applies unchanged from TQL-5. The supporting argument for the latter premise, given below, adds additional evidence to its TQL-5 analogue.

The new premises that apply at TQL-4 are:

4. Adequate planning has been conducted (AdqPlanningTQL4).
5. Adequate outputs of tool testing have been achieved (AdqTestingTQL4).

The supporting argument for these new premises is given below.

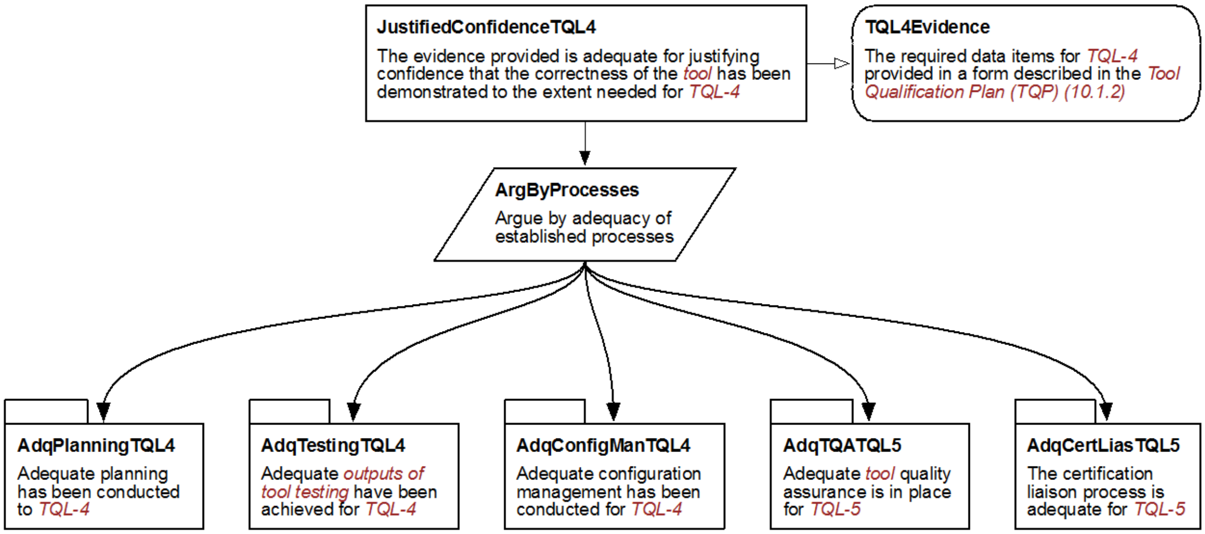

JustifiedConfidenceTQL4
The evidence provided is adequate for justifying confidence that the correctness of the tool has been demonstrated to the extent needed for TQL-4
TQL4Evidence
The required data items for TQL-4 provided in a form described in the Tool Qualification Plan (TQP) (10.1.2)
ArgByProcesses
Argue by adequacy of established processes
AdqPlanningTQL4
Adequate planning has been conducted to TQL-4
AdqTestingTQL4
Adequate outputs of tool testing have been achieved for TQL-4
AdqConfigManTQL4
Adequate configuration management has been conducted for TQL-4
AdqTQATQL5
Adequate tool quality assurance is in place for TQL-5
AdqCertLiasTQL5
The certification liaison process is adequate for TQL-5

**Figure B-18. JustifiedConfidenceTQL4**

Figure B-19 depicts the argument supporting the conclusion that “adequate planning has been conducted to TQL-4.” The argument is analogous to the DO-178C argument supporting the conclusion that “adequate planning has been conducted for Level D” (AdqPlanningLevD). There are two differences, both reflecting the differences in certification context between DO-178C and DO-330. First, whereas the DO-178C argument is concerned with the adequacy of the planning given the system into which the airborne software will be embedded, the DO-330 argument is concerned with the adequacy of planning given the airborne software the tool will be used to develop. Second, DO-330 defines a different set of “additional considerations” than DO-178C. Some of the DO-330 additional considerations, such as qualifying COTS tools, have DO-178C analogues. Others, such as multi-function tools, do not.

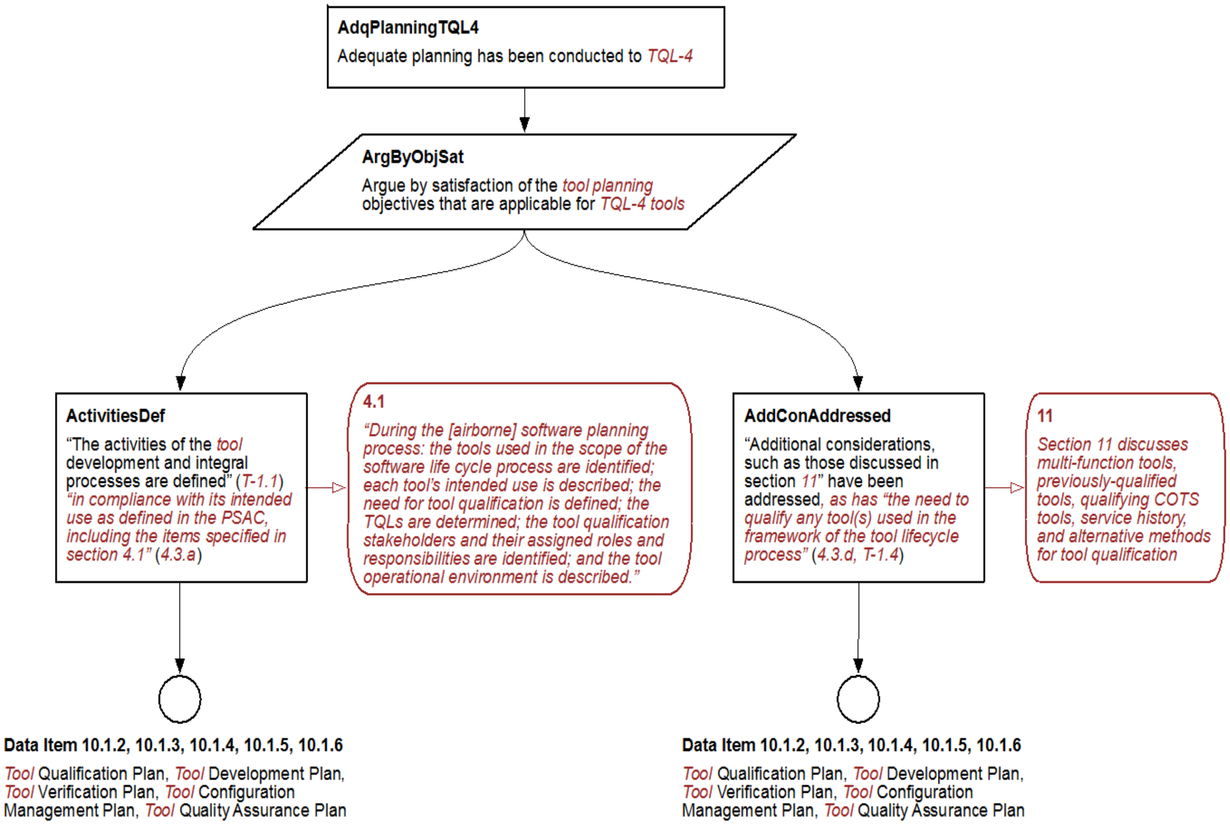

AdqPlanningTQL4
Adequate planning has been conducted to TQL-4
ArgByObjSat
Argue by satisfaction of the tool planning objectives that are applicable for TQL-4 tools
ActivitiesDef
“The activities of the tool development and integral processes are defined” (T-1.1) “in compliance with its intended use as defined in the PSAC, including the items specified in section 4.1” (4.3.a)
4.1
“During the [airborne] software planning process: the tools used in the scope of the software life cycle process are identified; each tool’s intended use is described; the need for tool qualification is defined; the TQLs are determined; the tool qualification stakeholders and their assigned roles and responsibilities are identified; and the tool operational environment is described.”
AddConAddressed
“Additional considerations, such as those discussed in section 11” have been addressed, as has “the need to qualify any tool(s) used in the framework of the tool lifecycle process” (4.3.d, T-1.4)
11
Section 11 discusses multi-function tools, previously-qualified tools, qualifying COTS tools, service history, and alternative methods for tool qualification
Data Item 10.1.2, 10.1.3, 10.1.4, 10.1.5, 10.1.6
Tool Qualification Plan, Tool Development Plan, Tool Verification Plan, Tool Configuration Management Plan, Tool Quality Assurance Plan
Data Item 10.1.2, 10.1.3, 10.1.4, 10.1.5, 10.1.6
Tool Qualification Plan, Tool Development Plan, Tool Verification Plan, Tool Configuration Management Plan, Tool Quality Assurance Plan

**Figure B-19. AdqPlanningTQL4**

Figure B-20 depicts the argument supporting the conclusion that "adequate planning has been conducted to TQL-4." The argument is analogous to the DO-178C argument supporting the conclusion that "sufficient verification of verification results [has] been achieved for Level D" (AdqVerVerResLevD). The main difference is that at TQL-4, DO-330 requires evidence that the test plan achieves coverage of the tool requirements. The DO-178C analogue is required only at Level C and above, at which level data coupling, control coupling, and statement coverage is also required.

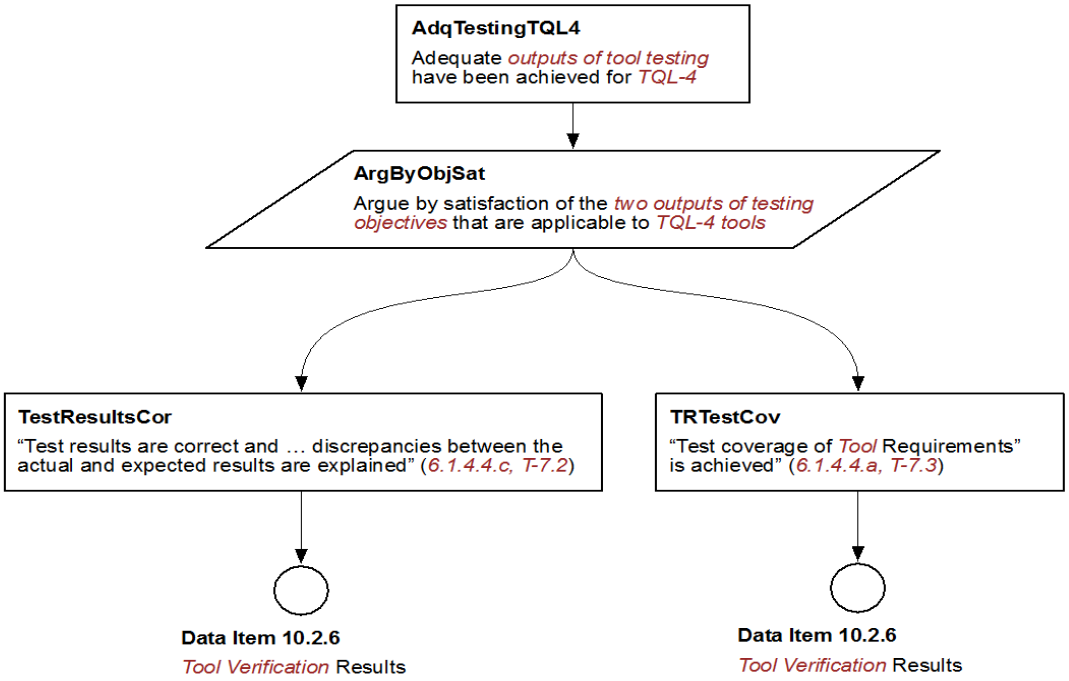

AdqTestingTQL4
Adequate outputs of tool testing have been achieved for TQL-4
ArgByObjSat
Argue by satisfaction of the two outputs of testing objectives that are applicable to TQL-4 tools
TestResultsCor
"Test results are correct and … discrepancies between the actual and expected results are explained" (6.1.4.4.c, T-7.2)
TRTestCov
"Test coverage of Tool Requirements" is achieved" (6.1.4.4.a, T-7.3)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-20. AdqTestingTQL4**

Figure B-21 depicts the argument supporting the conclusion that "adequate configuration management is in place for TQL-4." This argument extends a similar argument for TQL-5 (see figure B-10) and is largely analogous to the DO-178C argument supporting the conclusion that "adequate configuration management is in place for Level D" (AdqConfigManLevD). To the evidence required at TQL-5, this argument adds evidence showing that configuration baselines have been defined (BaseTraceEst); that problem reporting, change control, change review, and configuration status accounting have been established (ProbRepEtAllEst); and that environmental control has been established (EvnControlEst).

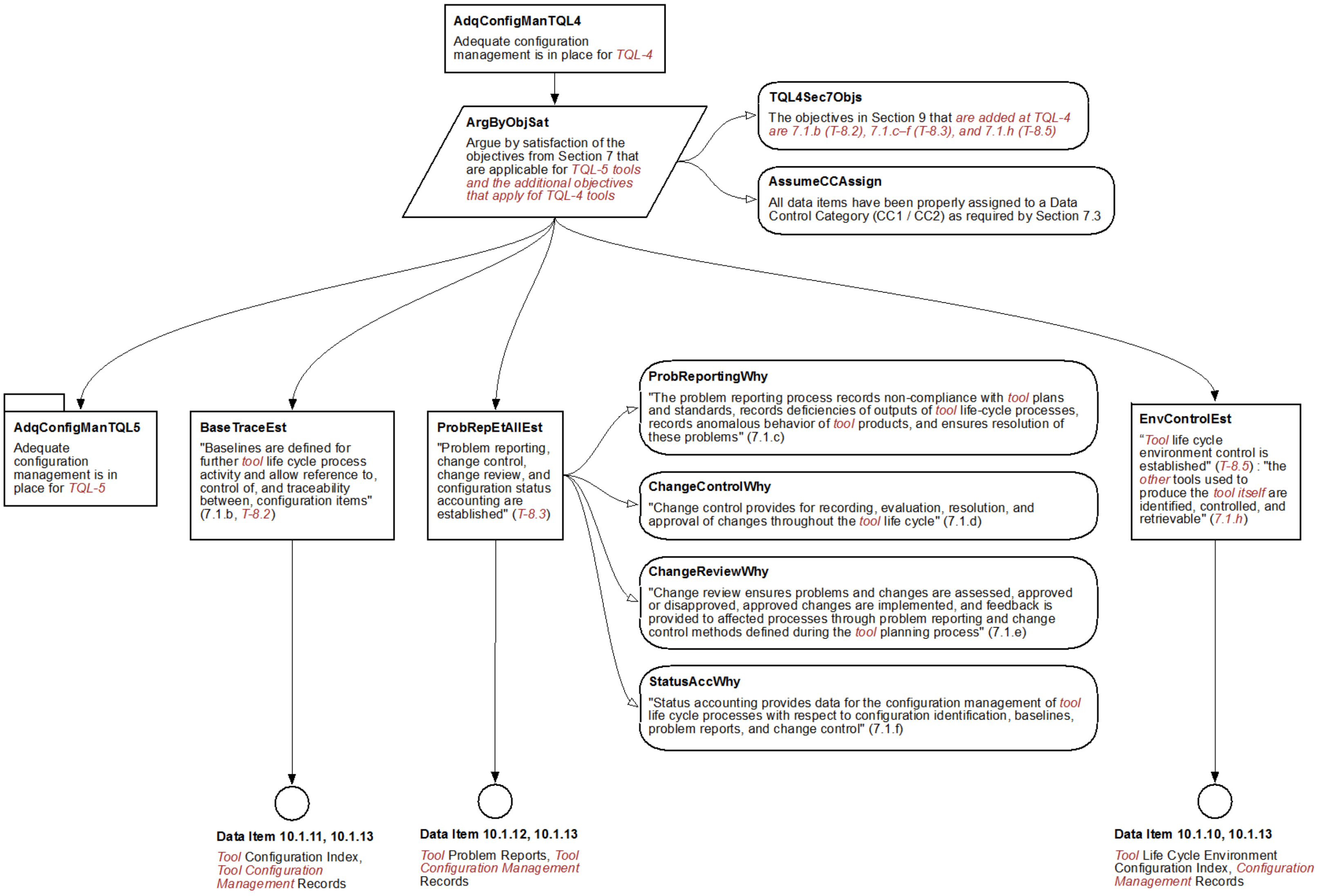

AdqConfigManTQL4
Adequate configuration management is in place for TQL-4
ArgByObjSat
Argue by satisfaction of the objectives from Section 7 that are applicable for TQL-5 tools and the additional objectives that apply for TQL-4 tools
TQL4Sec7Objs
The objectives in Section 9 that are added at TQL-4 are 7.1.b (T-8.2), 7.1.c–f (T-8.3), and 7.1.h (T-8.5)
AssumeCCAssign
All data items have been properly assigned to a Data Control Category (CC1 / CC2) as required by Section 7.3
AdqConfigManTQL5
Adequate configuration management is in place for TQL-5
BaseTraceEst
"Baselines are defined for further tool life cycle process activity and allow reference to, control of, and traceability between, configuration items" (7.1.b, T-8.2)
ProbRepEtAllEst
"Problem reporting, change control, change review, and configuration status accounting are established" (T-8.3)
ProbReportingWhy
"The problem reporting process records non-compliance with tool plans and standards, records deficiencies of outputs of tool life-cycle processes, records anomalous behavior of tool products, and ensures resolution of these problems" (7.1.c)
ChangeControlWhy
"Change control provides for recording, evaluation, resolution, and approval of changes throughout the tool life cycle" (7.1.d)
ChangeReviewWhy
"Change review ensures problems and changes are assessed, approved or disapproved, approved changes are implemented, and feedback is provided to affected processes through problem reporting and change control methods defined during the tool planning process" (7.1.e)
StatusAccWhy
"Status accounting provides data for the configuration management of tool life cycle processes with respect to configuration identification, baselines, problem reports, and change control" (7.1.f)
EnvControlEst
"Tool life cycle environment control is established" (T-8.5) : "the other tools used to produce the tool itself are identified, controlled, and retrievable" (7.1.h)
Data Item 10.1.11, 10.1.13
Tool Configuration Index, Tool Configuration Management Records
Data Item 10.1.12, 10.1.13
Tool Problem Reports, Tool Configuration Management Records
Data Item 10.1.10, 10.1.13
Tool Life Cycle Environment Configuration Index, Configuration Management Records

**Figure B-21. AdqConfigManTQL4B.5. Tool-Qualification Argument for TQL-3**

Figure B-22 depicts the argument supporting the conclusion that a tool "performs its intended function at an acceptable level of confidence for TQL-3." This argument is similar to the tool-qualification argument for TQL-4 and the DO-178C arguments in support of the conclusion that airborne software performs its acceptable level of safety at the appropriate software level.

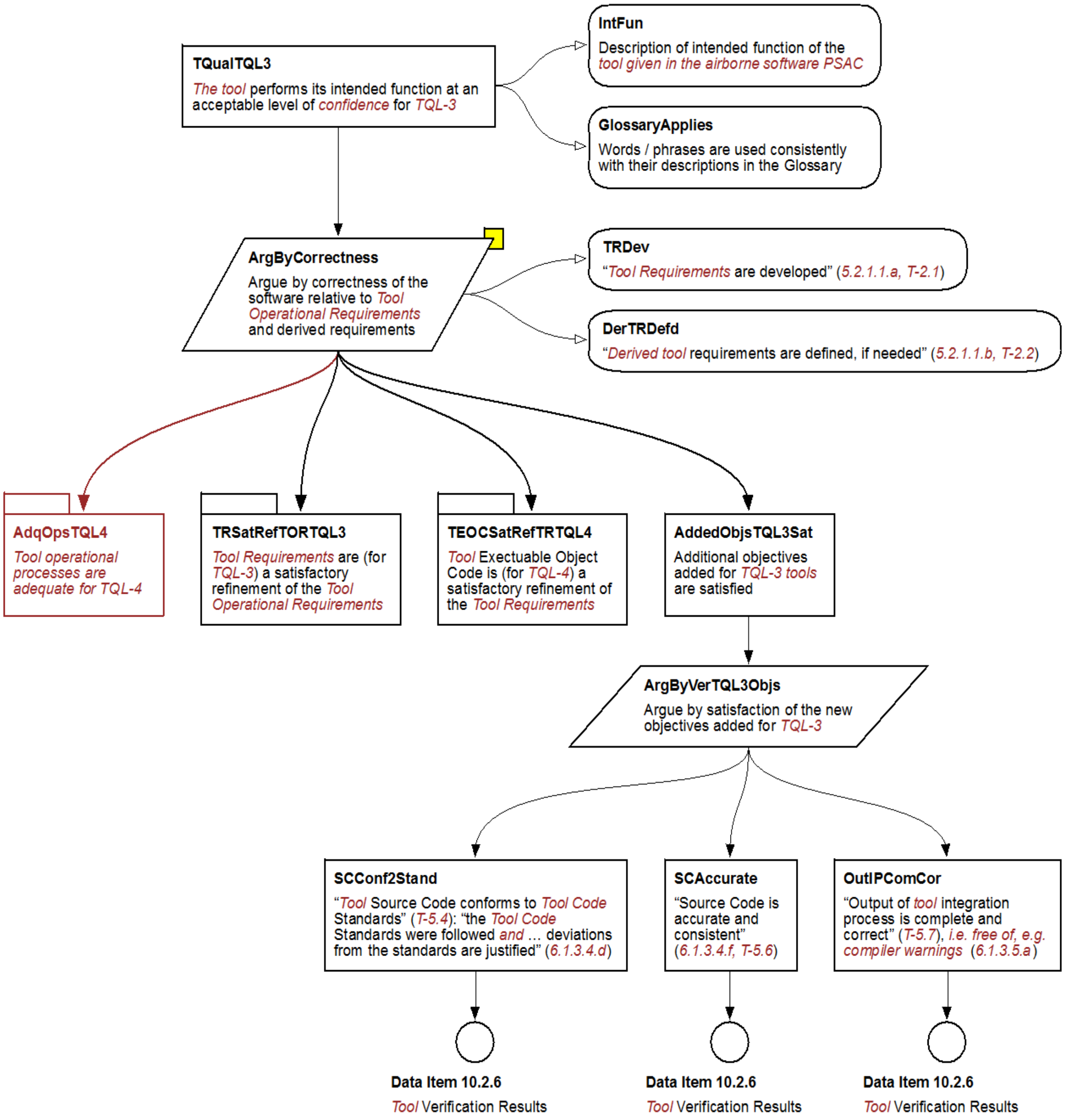

TQualTQL3
The tool performs its intended function at an acceptable level of confidence for TQL-3
IntFun
Description of intended function of the tool given in the airborne software PSAC
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
ArgByCorrectness
Argue by correctness of the software relative to Tool Operational Requirements and derived requirements
TRDev
"Tool Requirements are developed" (5.2.1.1.a, T-2.1)
DerTRDefd
"Derived tool requirements are defined, if needed" (5.2.1.1.b, T-2.2)
AdqOpsTQL4
Tool operational processes are adequate for TQL-4
TRSatRefTORTQL3
Tool Requirements are (for TQL-3) a satisfactory refinement of the Tool Operational Requirements
TEOCSatRefTRTQL4
Tool Exectuable Object Code is (for TQL-4) a satisfactory refinement of the Tool Requirements
AddedObjsTQL3Sat
Additional objectives added for TQL-3 tools are satisfied
ArgByVerTQL3Objs
Argue by satisfaction of the new objectives added for TQL-3
SCConf2Stand
"Tool Source Code conforms to Tool Code Standards" (T-5.4): "the Tool Code Standards were followed and ... deviations from the standards are justified" (6.1.3.4.d)
SCAccurate
"Source Code is accurate and consistent" (6.1.3.4.f, T-5.6)
OutIPComCor
"Output of tool integration process is complete and correct" (T-5.7), i.e. free of, e.g. compiler warnings (6.1.3.5.a)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-22. TQualTQL3**

This argument revises the TQL-4 argument’s premise about the adequacy of tool requirements to reflect the higher tool qualification level (TRSatRefTORTQL3). It also adds new premises to reflect three source-code related objectives that are added at TQL-3:

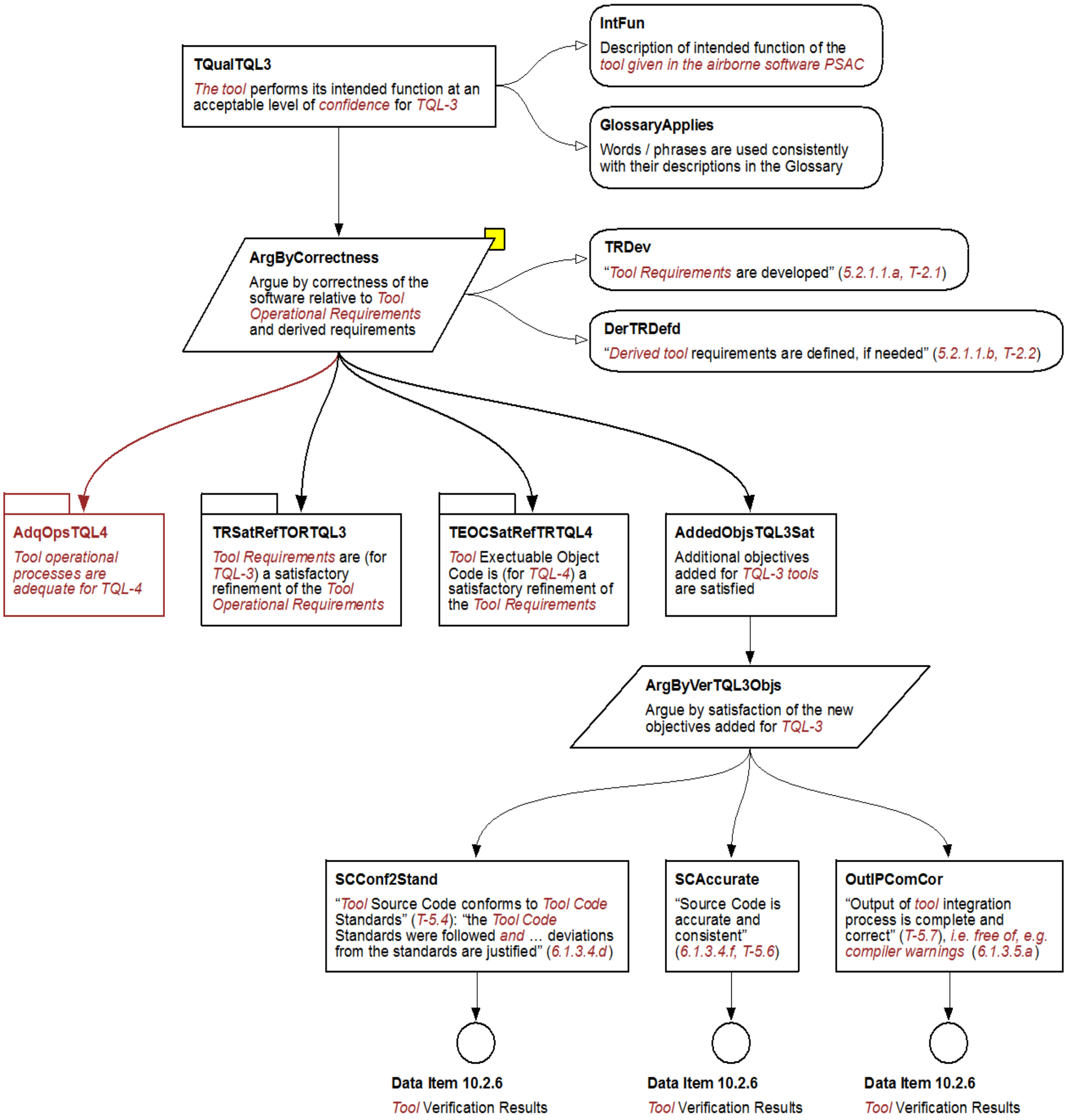
1. The tool source code conforms to the tool code standards, except where deviations are justified (SCConf2Stand).
2. The (tool) source code is accurate and consistent (SCAccurate).
3. The output of the tool integration is complete and correct (OutIPComCor).

These new premises have analogues in the DO-178C argument supporting the conclusion that the source code and related outputs are satisfactory for Level C (SCSatLevC).

Section B.3.1 presents the argument that the tool requirements are a satisfactory refinement of the tool operational requirements. Section B.3.presents the argument for confidence in the TQL-3 tool-qualification argument.

## B 5. TOOL QUALIFICATION ARGUMENT FOR TQL-3

Figure B-23 depicts the argument supporting the conclusion that a tool "performs its intended function at an acceptable level of confidence for TQL-3." This argument is similar to the tool qualification argument for TQL-4 and the DO-178C arguments in support of the conclusion that airborne software performs its acceptable level of safety at the appropriate software level.



TQualTQL3
The tool performs its intended function at an acceptable level of confidence for TQL-3
IntFun
Description of intended function of the tool given in the airborne software PSAC
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
ArgByCorrectness
Argue by correctness of the software relative to Tool Operational Requirements and derived requirements
TRDev
"Tool Requirements are developed" (5.2.1.1.a, T-2.1)
DerTRDefd
"Derived tool requirements are defined, if needed" (5.2.1.1.b, T-2.2)
AdqOpsTQL4
Tool operational processes are adequate for TQL-4
TRSatRefTORTQL3
Tool Requirements are (for TQL-3) a satisfactory refinement of the Tool Operational Requirements
TEOCSatRefTRTQL4
Tool Exectuable Object Code is (for TQL-4) a satisfactory refinement of the Tool Requirements
AddedObjsTQL3Sat
Additional objectives added for TQL-3 tools are satisfied
ArgByVerTQL3Objs
Argue by satisfaction of the new objectives added for TQL-3
SCConf2Stand
"Tool Source Code conforms to Tool Code Standards" (T-5.4): "the Tool Code Standards were followed and ... deviations from the standards are justified" (6.1.3.4.d)
SCAccurate
"Source Code is accurate and consistent" (6.1.3.4.f, T-5.6)
OutIPComCor
"Output of tool integration process is complete and correct" (T-5.7), i.e. free of, e.g. compiler warnings (6.1.3.5.a)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-23. TQualTQL3**

This argument revises the TQL-4 argument's premise about tool requirements adequacy to reflect the higher tool qualification level (TRSatRefTORTQL3). It also adds new premises to reflect three source-code related objectives that are added at TQL-3:

1. The Tool Source Code conforms to the Tool Code Standards except where deviations are justified (SCConf2Stand)
2. The (Tool) Source Code is accurate and consistent (SCAccurate)
3. The output of the tool integration is complete and correct (OutIPComCor)

These new premises have analogues in the DO-178C argument supporting the conclusion that the Source Code and related outputs are satisfactory for Level C (SCSatLevC).

Section B 5.1 gives the argument that the Tool Requirements are a satisfactory refinement of the Tool Operational Requirements. Section B 5.2 gives the argument for confidence in the TQL-3 tool qualification argument.

### B.5.1. EVIDENCE OF CORRECTNESS FOR TQL-3

Figure B-22 presents the argument that the tool performs its intended function at an acceptable level of confidence for TQL-3. This functionality is inferred from the tool's correctness with respect to its tool operational requirements, which is in turn inferred from several premises. Among these is the proposition that "tool requirements are (for TQL-3) a satisfactory refinement of the tool operational requirements" (TRSatRefTORTQL3). Figure B-24 presents the argument supporting that proposition. The argument adds an additional premise to the analogous TQL-4 argument (see figure B-16), namely that the "Tool Requirements Standards were followed during the tool requirements process and deviations from the standards are justified" (TRSatRefTORTQL3).

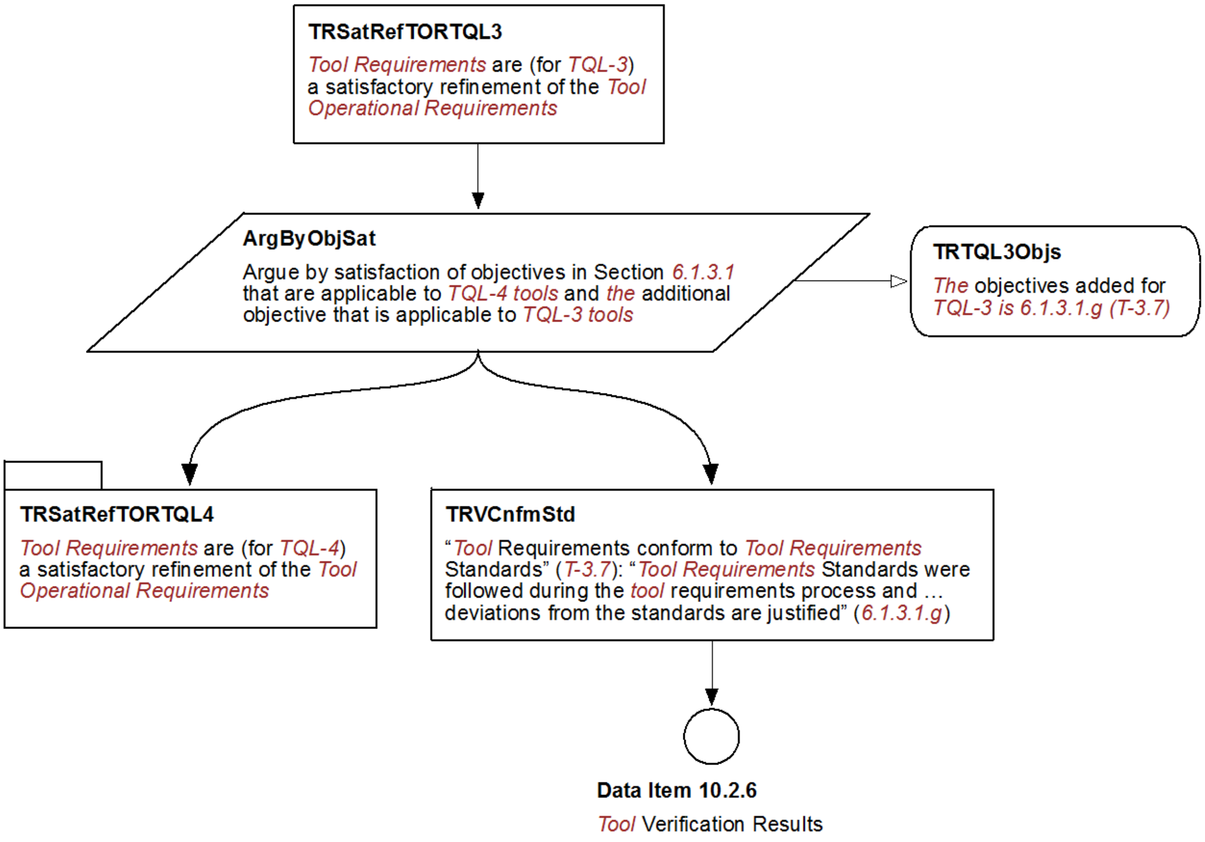

TRSatRefTORTQL3
Tool Requirements are (for TQL-3) a satisfactory refinement of the Tool Operational Requirements
ArgByObjSat
Argue by satisfaction of objectives in Section 6.1.3.1 that are applicable to TQL-4 tools and the additional objective that is applicable to TQL-3 tools
TRTQL3Objs
The objectives added for TQL-3 is 6.1.3.1.g (T-3.7)
TRSatRefTORTQL4
Tool Requirements are (for TQL-4) a satisfactory refinement of the Tool Operational Requirements
TRVCnfmStd
"Tool Requirements conform to Tool Requirements Standards" (T-3.7): "Tool Requirements Standards were followed during the tool requirements process and … deviations from the standards are justified" (6.1.3.1.g)
Data Item 10.2.6
Tool Verification Results

**Figure B-24. TRSatRefTORTQL3**

## B.5.2. CONFIDENCE IN TOOL QUALIFICATION ARGUMENT FOR TQL-3

As in the DO-178C argument, confidence that demonstrating that software is correct with respect to its requirements is sufficient to show that the software will perform its intended function would be undermined by the use of inadequate processes. Figure B-25 depicts the confidence argument for TQL-3. The argument infers that “the evidence provided is adequate for justifying that the correctness of the tool has been demonstrated to the extent needed for TQL-3” on the grounds of five premises applied at TQL-4 and one additional premise. The existing premises are:

1. Adequate planning has been conducted (AdqPlanningTQL3).
2. Adequate outputs of tool testing have been achieved (AdqTestingTQL3).
3. Adequate tool quality assurance is in place (AdqTQATQL3).
4. Adequate configuration management has been conducted (AdqConfigManTQL4).
5. The certification liaison process is adequate (AdqCertLiasTQL5).

The supporting argument for the latter two premises applies unchanged from TQL-4. The supporting arguments for the former three premises, given below, add additional evidence to their TQL-4 analogues.

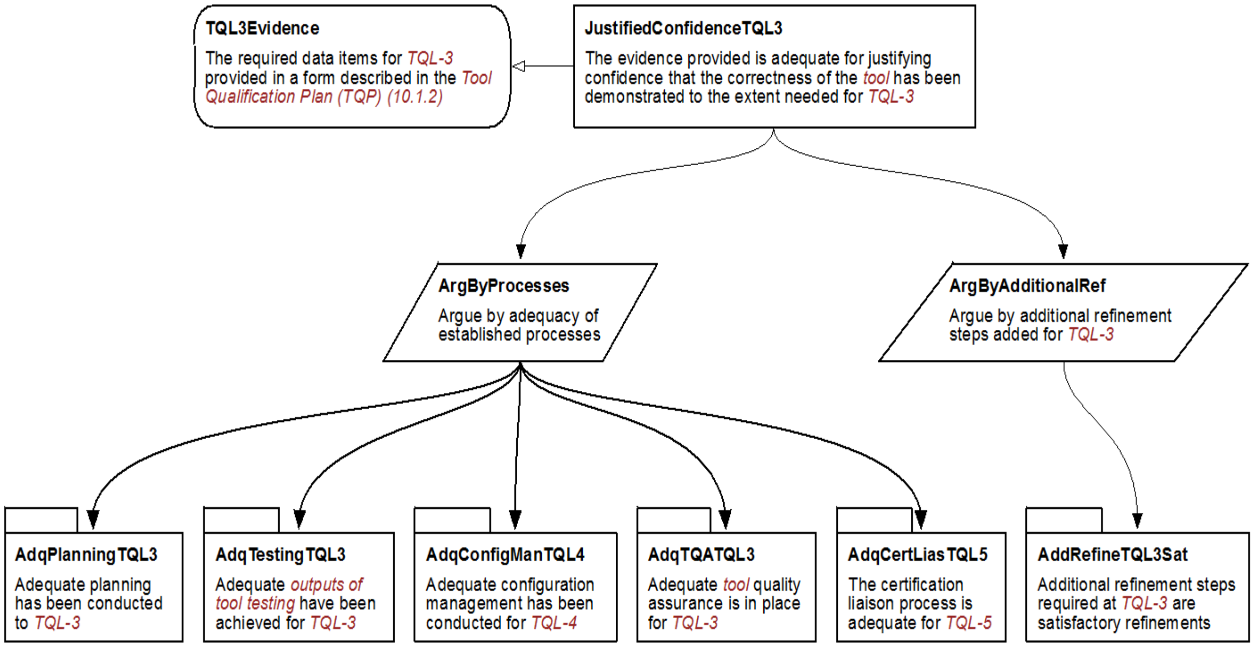

TQL3Evidence
The required data items for TQL-3 provided in a form described in the Tool Qualification Plan (TQP) (10.1.2)
JustifiedConfidenceTQL3
The evidence provided is adequate for justifying confidence that the correctness of the tool has been demonstrated to the extent needed for TQL-3
ArgByProcesses
Argue by adequacy of established processes
ArgByAdditionalRef
Argue by additional refinement steps added for TQL-3
AdqPlanningTQL3
Adequate planning has been conducted to TQL-3
AdqTestingTQL3
Adequate outputs of tool testing have been achieved for TQL-3
AdqConfigManTQL4
Adequate configuration management has been conducted for TQL-4
AdqTQATQL3
Adequate tool quality assurance is in place for TQL-3
AdqCertLiasTQL5
The certification liaison process is adequate for TQL-5
AddRefineTQL3Sat
Additional refinement steps required at TQL-3 are satisfactory refinements

**Figure B-25. JustifiedConfidenceTQL3**

The premise that has no TQL-4 analogue is that “additional refinement steps required at TQL-3 are satisfactory” (AddRefineTQL3Sat). The supporting argument for this premise follows.

Figure B-26 depicts the argument supporting the conclusion that “adequate planning has been conducted to TQL-3. This argument extends the analogous argument for TQL-4 (see figure B-19) and is analogous to DO-178C's Level C adequate planning argument (AdqPlanningLevC). The argument adds five new evidence-supported premises to its TQL-4 analogue:

1. The tool's life cycle is defined (LifeCycleDef).
2. The tool-development environment is selected and defined (LifeCycleEnv).
3. Tool development standards have been defined (ToolDevStds).
4. Tool plans comply with DO-330 §10, which describes typical and minimum content for tool-qualification life cycle data items (ToolPlansComply).
5. Development and revision of tool plans are coordinated (DevRevCoord).

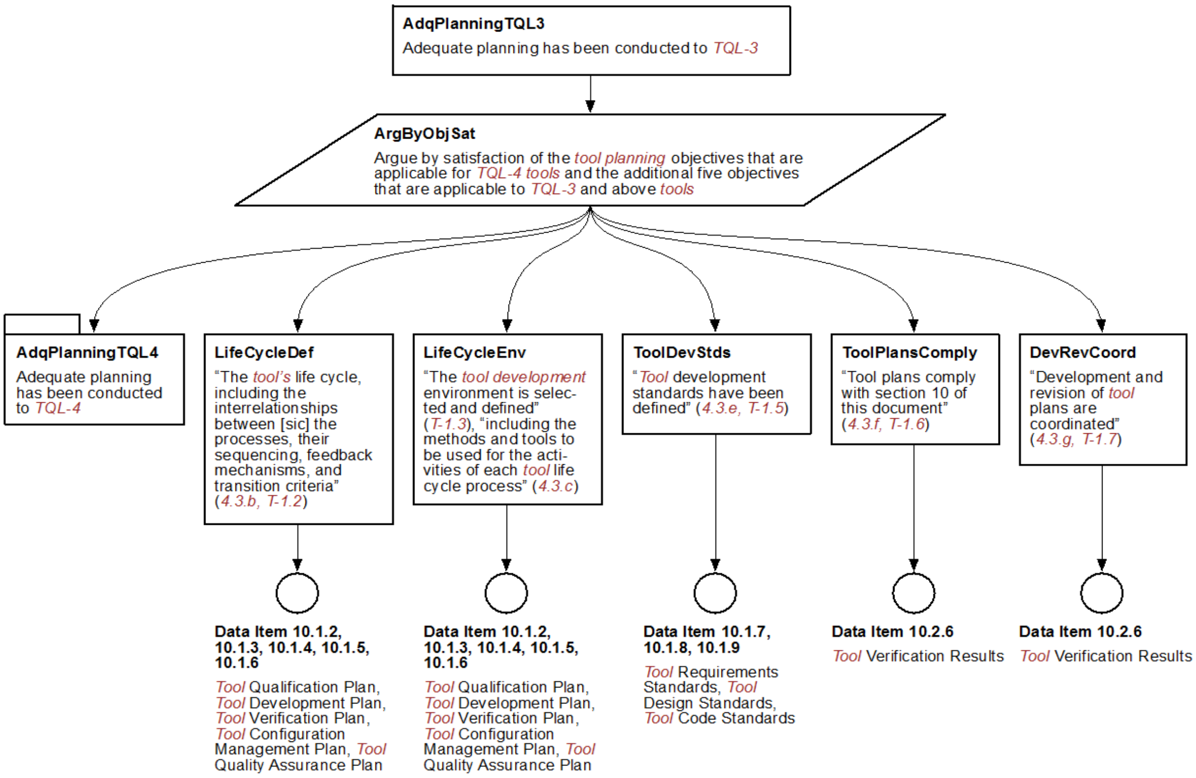

AdqPlanningTQL3
Adequate planning has been conducted to TQL-3
ArgByObjSat
Argue by satisfaction of the tool planning objectives that are applicable for TQL-4 tools and the additional five objectives that are applicable to TQL-3 and above tools
AdqPlanningTQL4
Adequate planning has been conducted to TQL-4
LifeCycleDef
"The tool's life cycle, including the interrelationships between [sic] the processes, their sequencing, feedback mechanisms, and transition criteria" (4.3.b, T-1.2)
LifeCycleEnv
"The tool development environment is selected and defined" (T-1.3), "including the methods and tools to be used for the activities of each tool life cycle process" (4.3.c)
ToolDevStds
"Tool development standards have been defined" (4.3.e, T-1.5)
ToolPlansComply
"Tool plans comply with section 10 of this document" (4.3.f, T-1.6)
DevRevCoord
"Development and revision of tool plans are coordinated" (4.3.g, T-1.7)
Data Item 10.1.2, 10.1.3, 10.1.4, 10.1.5, 10.1.6
Tool Qualification Plan, Tool Development Plan, Tool Verification Plan, Tool Configuration Management Plan, Tool Quality Assurance Plan
Data Item 10.1.2, 10.1.3, 10.1.4, 10.1.5, 10.1.6
Tool Qualification Plan, Tool Development Plan, Tool Verification Plan, Tool Configuration Management Plan, Tool Quality Assurance Plan
Data Item 10.1.7, 10.1.8, 10.1.9
Tool Requirements Standards, Tool Design Standards, Tool Code Standards
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-26. AdqPlanningTQL3**

Figure B-27 depicts the argument supporting the conclusion that "adequate outputs of tool testing have been achieved for TQL-3." This argument extends the analogous argument for TQL-4 (see figure B-20) and is analogous to DO-178C's Level C sufficient verification argument (AdqVerResVerLevC). The argument adds four new evidence-supported premises to its TQL-4 analogue:

1. The test cases were correctly developed into test procedures (TestProcCorr).
2. The test cases cover the low-level tool requirements (TRTestCovLLTR).
3. The test cases achieve statement coverage of the tool source code (StatementCov).
4. The test cases achieve coverage of data and control coupling (TestCovCoupling).

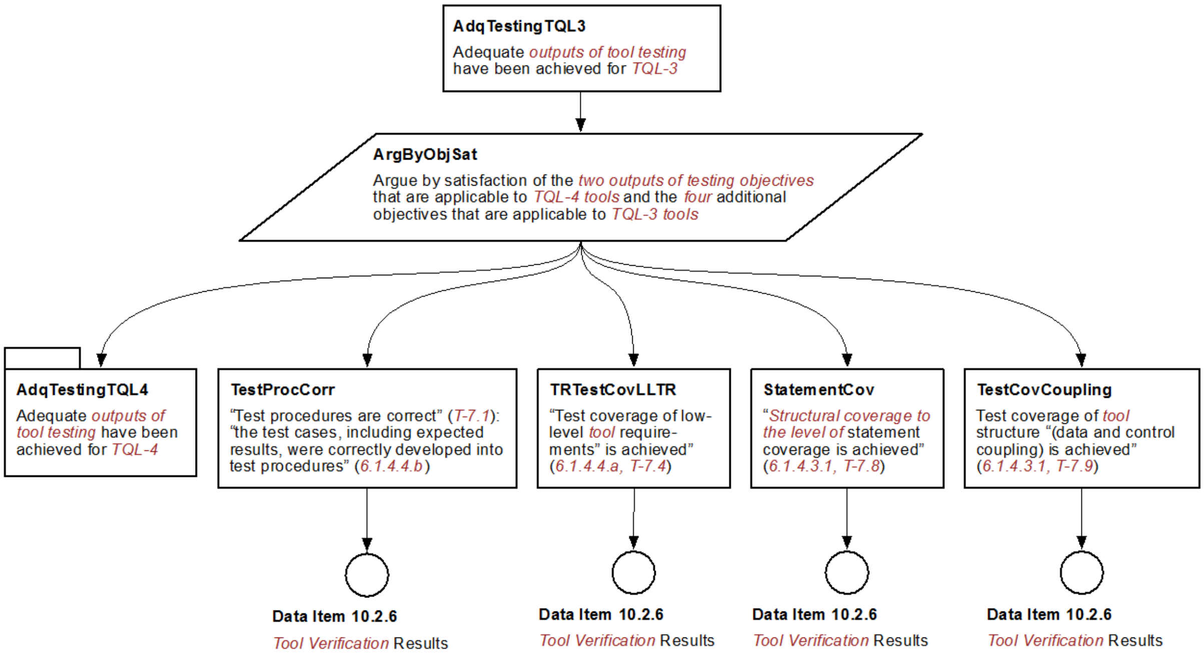

AdqTestingTQL3
Adequate outputs of tool testing have been achieved for TQL-3
ArgByObjSat
Argue by satisfaction of the two outputs of testing objectives that are applicable to TQL-4 tools and the four additional objectives that are applicable to TQL-3 tools
AdqTestingTQL4
Adequate outputs of tool testing have been achieved for TQL-4
TestProcCorr
"Test procedures are correct" (T-7.1): "the test cases, including expected results, were correctly developed into test procedures" (6.1.4.4.b)
TRTestCovLLTR
"Test coverage of low-level tool requirements" is achieved" (6.1.4.4.a, T-7.4)
StatementCov
"Structural coverage to the level of statement coverage is achieved" (6.1.4.3.1, T-7.8)
TestCovCoupling
Test coverage of tool structure "(data and control coupling) is achieved" (6.1.4.3.1, T-7.9)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-27. AdqTestingTQL3**

Figure B-28 depicts the argument supporting the conclusion that “adequate tool quality assurance is in place for TQL-3.” This argument extends the analogous argument for TQL-5 (see figure B-11) and is analogous to DO-178C’s Level C adequate software quality assurance argument (AdqSQALevC). The argument adds three new evidence-supported premises to its TQL-4 analogue:

1. Tool plans and standards are developed and reviewed for consistency (AssurePlansRev).
2. Tool development processes comply with approved tool standards (AssureCompStans).
3. Transition criteria for the tool life-cycle process are satisfied (AssureTransCrit).

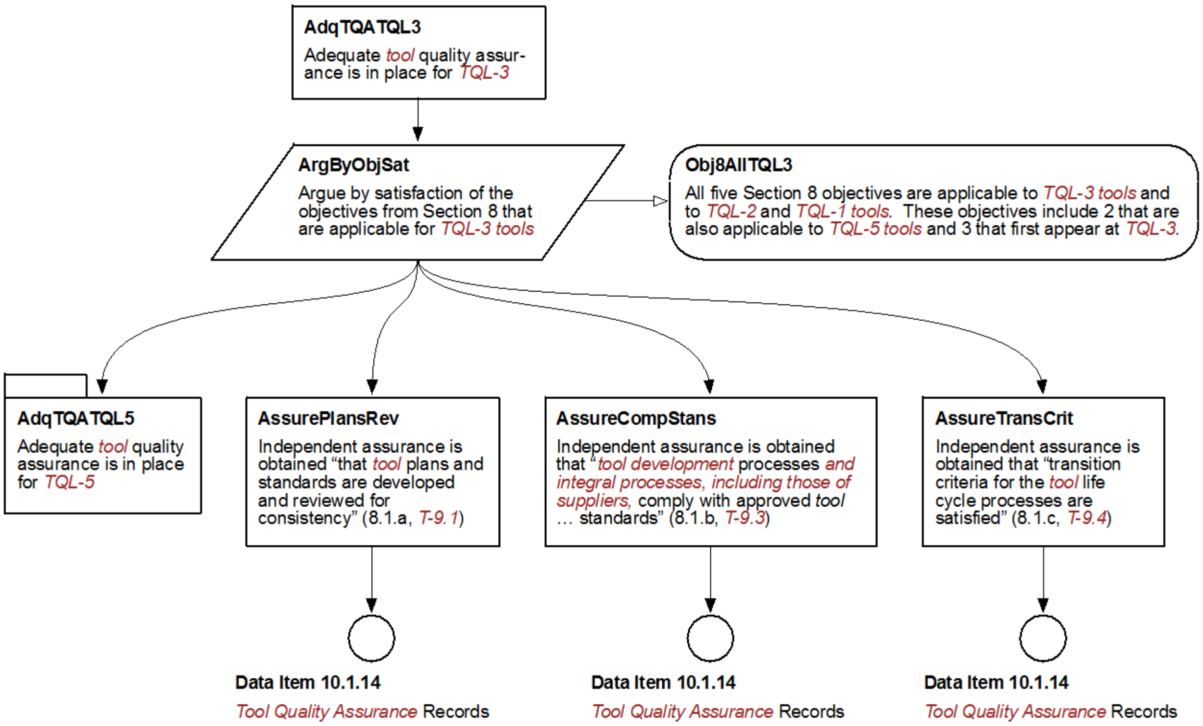

AdqTQATQL3
Adequate tool quality assurance is in place for TQL-3
ArgByObjSat
Argue by satisfaction of the objectives from Section 8 that are applicable for TQL-3 tools
Obj8AllTQL3
All five Section 8 objectives are applicable to TQL-3 tools and to TQL-2 and TQL-1 tools. These objectives include 2 that are also applicable to TQL-5 tools and 3 that first appear at TQL-3.
AdqTQATQL5
Adequate tool quality assurance is in place for TQL-5
AssurePlansRev
Independent assurance is obtained “that tool plans and standards are developed and reviewed for consistency” (8.1.a, T-9.1)
AssureCompStans
Independent assurance is obtained that “tool development processes and integral processes, including those of suppliers, comply with approved tool … standards” (8.1.b, T-9.3)
AssureTransCrit
Independent assurance is obtained that “transition criteria for the tool life cycle processes are satisfied” (8.1.c, T-9.4)
Data Item 10.1.14
Tool Quality Assurance Records
Data Item 10.1.14
Tool Quality Assurance Records
Data Item 10.1.14
Tool Quality Assurance Records

**Figure B-28. AdqTQATQL3**

Figure B-29 depicts the argument supporting the conclusion that "additional refinement steps required at TQL-3 are satisfactory refinements." This argument has no analogue at lower TQLs. The three additional refinement steps are:

1. The low-level tool requirements refine the tool requirements (LLRSatTQL3).
2. The tool source code and related outputs are satisfactory (TSCSatTQL3).
3. The tool executable object code refines the low-level tool requirements (TEOCSatLLTRTQL3).

The arguments supporting each of these premises are presented below.

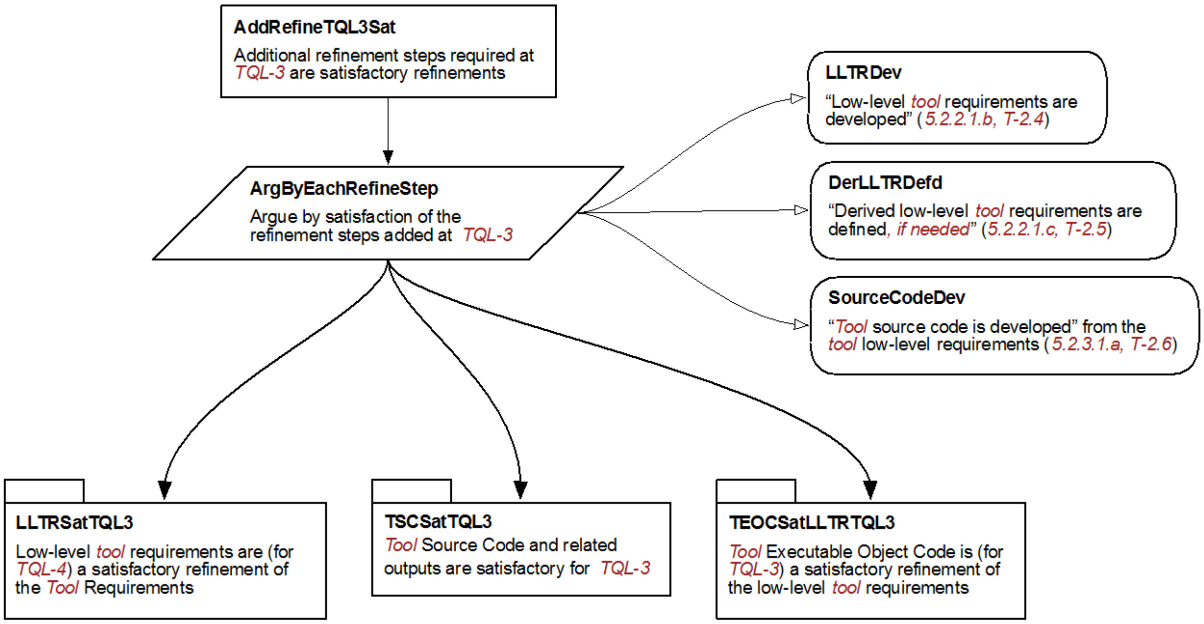

AddRefineTQL3Sat
Additional refinement steps required at TQL-3 are satisfactory refinements
ArgByEachRefineStep
Argue by satisfaction of the refinement steps added at TQL-3
LLTRDev
"Low-level tool requirements are developed" (5.2.2.1.b, T-2.4)
DerLLTRDefd
"Derived low-level tool requirements are defined, if needed" (5.2.2.1.c, T-2.5)
SourceCodeDev
"Tool source code is developed" from the tool low-level requirements (5.2.3.1.a, T-2.6)
LLTRSatTQL3
Low-level tool requirements are (for TQL-4) a satisfactory refinement of the Tool Requirements
TSCSatTQL3
Tool Source Code and related outputs are satisfactory for TQL-3
TEOCSatLLTRTQL3
Tool Executable Object Code is (for TQL-3) a satisfactory refinement of the low-level tool requirements

**Figure B-29. AddRefineTQL3Sat**

Figure B-30 depicts the argument supporting the conclusion that the "low-level tool requirements are (for TQL-3) a satisfactory refinement of the Tool Requirements." This argument has no analogues at lower TQLs but is analogous to a combination of two DO-178C arguments: the argument showing that low-level requirements are satisfactory (LLRAdqLevelC) and the argument that the software architecture is satisfactory (SwArchAdqLevelC). The argument depends on eight evidence-supported premises:

1. The low-level tool requirements comply with the tool requirements (TLLRComply).
2. The low-level tool requirements are accurate and consistent (TLLRAccurate).
3. The low-level tool requirements conform to the tool design standards (LLTRConfStand).
4. The low-level tool requirements are traceable to the tool requirements (LLTRTraceTR).
5. The algorithms specified are accurate (AlgorAcc).
6. The tool architecture is compatible with the tool requirements (TArchCapatTR).
7. The tool architecture is (internally) consistent (TArchCnsstnt).
8. The tool architecture conforms to the tool design standards (TArchConforms).

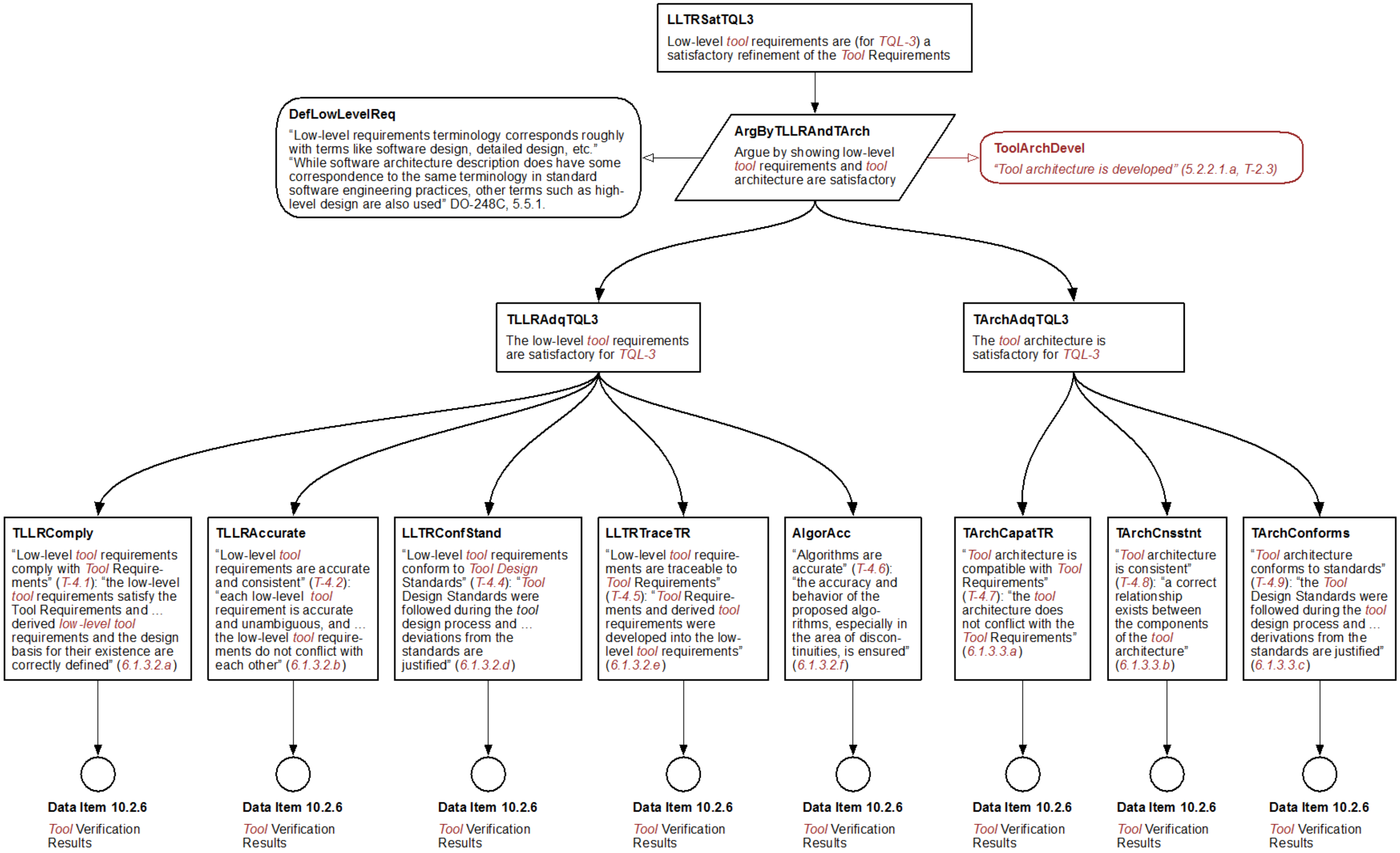
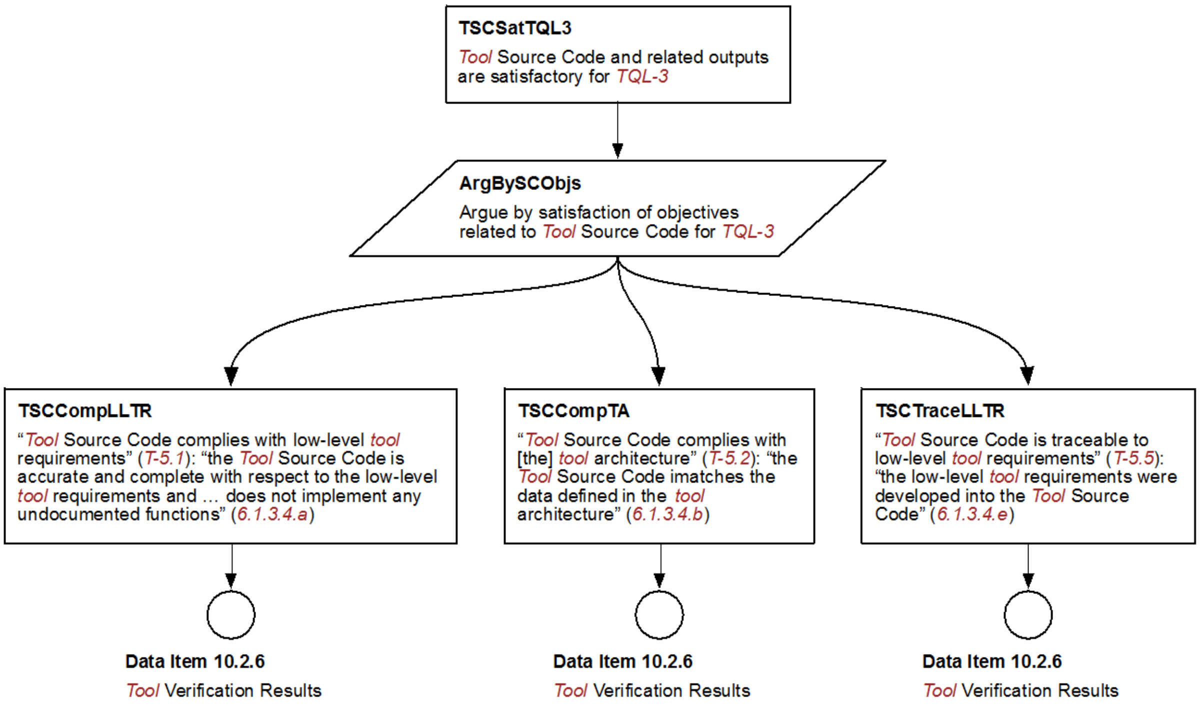

LLTRSatTQL3
Low-level tool requirements are (for TQL-3) a satisfactory refinement of the Tool Requirements
DefLowLevelReq
"Low-level requirements terminology corresponds roughly with terms like software design, detailed design, etc." "While software architecture description does have some correspondence to the same terminology in standard software engineering practices, other terms such as high-level design are also used" DO-248C, 5.5.1.
ArgByTLLRAndTArch
Argue by showing low-level tool requirements and tool architecture are satisfactory
ToolArchDevel
"Tool architecture is developed" (5.2.2.1.a, T-2.3)
TLLRAdqTQL3
The low-level tool requirements are satisfactory for TQL-3
TArchAdqTQL3
The tool architecture is satisfactory for TQL-3
TLLRComply
"Low-level tool requirements comply with Tool Requirements" (T-4.1): "the low-level tool requirements satisfy the Tool Requirements and ... derived low-level tool requirements and the design basis for their existence are correctly defined" (6.1.3.2.a)
TLLRAccurate
"Low-level tool requirements are accurate and consistent" (T-4.2): "each low-level tool requirement is accurate and unambiguous, and ... the low-level tool requirements do not conflict with each other" (6.1.3.2.b)
LLTRConfStand
"Low-level tool requirements conform to Tool Design Standards" (T-4.4): "Tool Design Standards were followed during the tool design process and ... deviations from the standards are justified" (6.1.3.2.d)
LLTRTraceTR
"Low-level tool requirements are traceable to Tool Requirements" (T-4.5): "Tool Requirements and derived tool requirements were developed into the low-level tool requirements" (6.1.3.2.e)
AlgorAcc
"Algorithms are accurate" (T-4.6): "the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities, is ensured" (6.1.3.2.f)
TArchCapatTR
"Tool architecture is compatible with Tool Requirements" (T-4.7): "the tool architecture does not conflict with the Tool Requirements" (6.1.3.3.a)
TArchCnsstnt
"Tool architecture is consistent" (T-4.8): "a correct relationship exists between the components of the tool architecture" (6.1.3.3.b)
TArchConforms
"Tool architecture conforms to standards" (T-4.9): "the Tool Design Standards were followed during the tool design process and ... derivations from the standards are justified" (6.1.3.3.c)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-30. LLTRSatTQL3**

Figure B-31 depicts the argument supporting the conclusion that the "Tool Source Code and related outputs are satisfactory for TQL-3." This argument has no analogue at lower TQLs but is analogous to part of DO-178C's argument that the airborne software source code and related outputs are satisfactory (SCSatLevC/SCmatchesDesign). The argument depends on three evidence-supported premises:

1. The tool source code complies with low-level tool requirements (TSCCompLLTR).
2. The tool source code complies with the tool architecture (TSCCompTA).
3. The tool source code is traceable to the low-level tool requirements (TSCTraceLLTR).



**TSCSatTQL3**

*Tool* Source Code and related outputs are satisfactory for *TQL-3*

**ArgBySCObjs**

Argue by satisfaction of objectives related to *Tool* Source Code for *TQL-3*

**TSCCompLLTR**

"*Tool* Source Code complies with low-level *tool* requirements" (*T-5.1*): "the *Tool* Source Code is accurate and complete with respect to the low-level *tool* requirements and … does not implement any undocumented functions" (*6.1.3.4.a*)

**TSCCompTA**

"*Tool* Source Code complies with [the] *tool* architecture" (*T-5.2*): "the *Tool* Source Code imatches the data defined in the *tool* architecture" (*6.1.3.4.b*)

**TSCTraceLLTR**

"*Tool* Source Code is traceable to low-level *tool* requirements" (*T-5.5*): "the low-level *tool* requirements were developed into the *Tool* Source Code" (*6.1.3.4.e*)

**Data Item 10.2.6**

*Tool* Verification Results

**Data Item 10.2.6**

*Tool* Verification Results

**Data Item 10.2.6**

*Tool* Verification Results

**Figure B-31. TSCSatTQL3**

Figure B-32 depicts the argument supporting the conclusion that the "Tool Executable Object Code is (for TQL-3) a satisfactory refinement of the low-level tool requirements." Although this argument has no analogue at lower TQLs, it is analogous to DO-178C's argument that the airborne software's Executable Object Code satisfactorily refines its low-level requirements (EOCSatLLLevC). The argument depends on two evidence-supported premises:

1. The tool executable object code complies with the low-level tool requirements (TEOCCompliesLLTR).
2. The tool executable object code is robust with the low-level tool requirements (TEOCRobustLLTR).

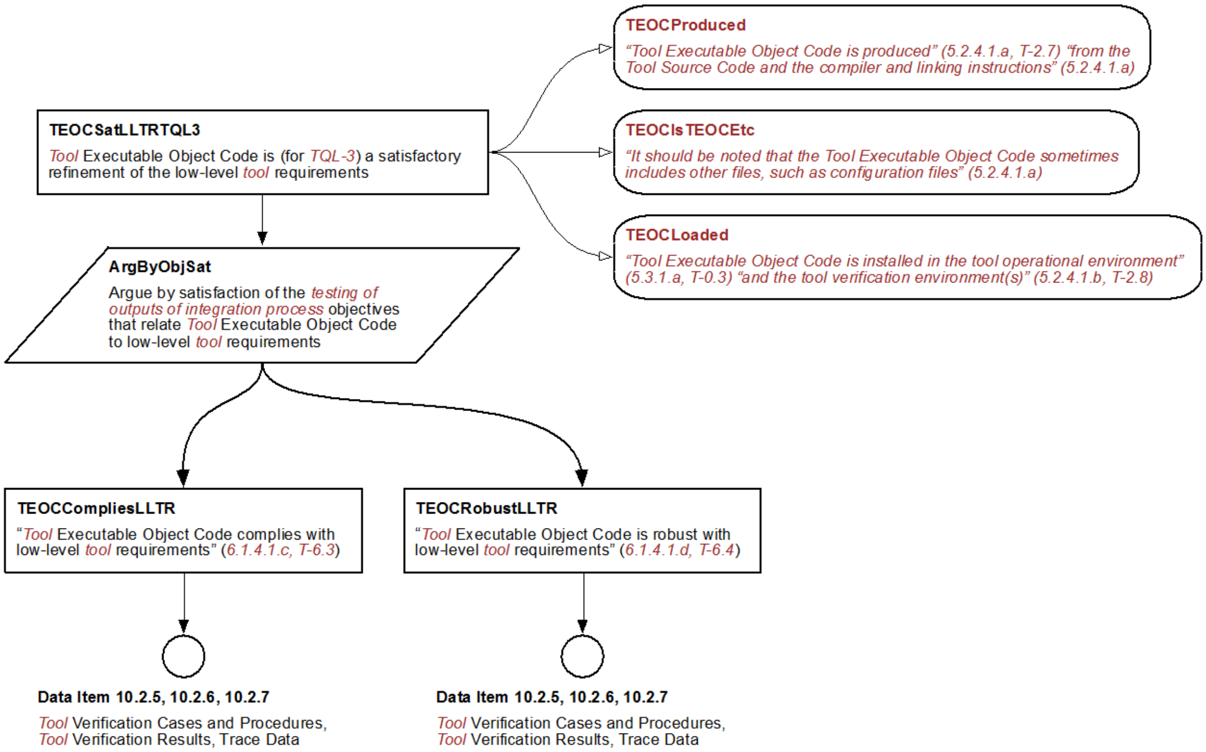

TEOCProduced
"Tool Executable Object Code is produced" (5.2.4.1.a, T-2.7) "from the Tool Source Code and the compiler and linking instructions" (5.2.4.1.a)
TEOCSatLLTRTQL3
Tool Executable Object Code is (for TQL-3) a satisfactory refinement of the low-level tool requirements
TEOCIsTEOCEtc
"It should be noted that the Tool Executable Object Code sometimes includes other files, such as configuration files" (5.2.4.1.a)
TEOCLoaded
"Tool Executable Object Code is installed in the tool operational environment" (5.3.1.a, T-0.3) "and the tool verification environment(s)" (5.2.4.1.b, T-2.8)
ArgByObjSat
Argue by satisfaction of the testing of outputs of integration process objectives that relate Tool Executable Object Code to low-level tool requirements
TEOCCompliesLLTR
"Tool Executable Object Code complies with low-level tool requirements" (6.1.4.1.c, T-6.3)
TEOCRobustLLTR
"Tool Executable Object Code is robust with low-level tool requirements" (6.1.4.1.d, T-6.4)
Data Item 10.2.5, 10.2.6, 10.2.7
Tool Verification Cases and Procedures, Tool Verification Results, Trace Data
Data Item 10.2.5, 10.2.6, 10.2.7
Tool Verification Cases and Procedures, Tool Verification Results, Trace Data

**Figure B-32. TEOCSatLLTRTQL3**

## B.6. TOOL QUALIFICATION ARGUMENT FOR TQL-2

Figure B-33 depicts the argument supporting the conclusion that a tool "performs its intended function at an acceptable level of confidence for TQL-2." This argument is similar to the tool-qualification argument for TQL-3 and the DO-178C arguments in support of the conclusion that airborne software performs its acceptable level of safety at the appropriate software level.

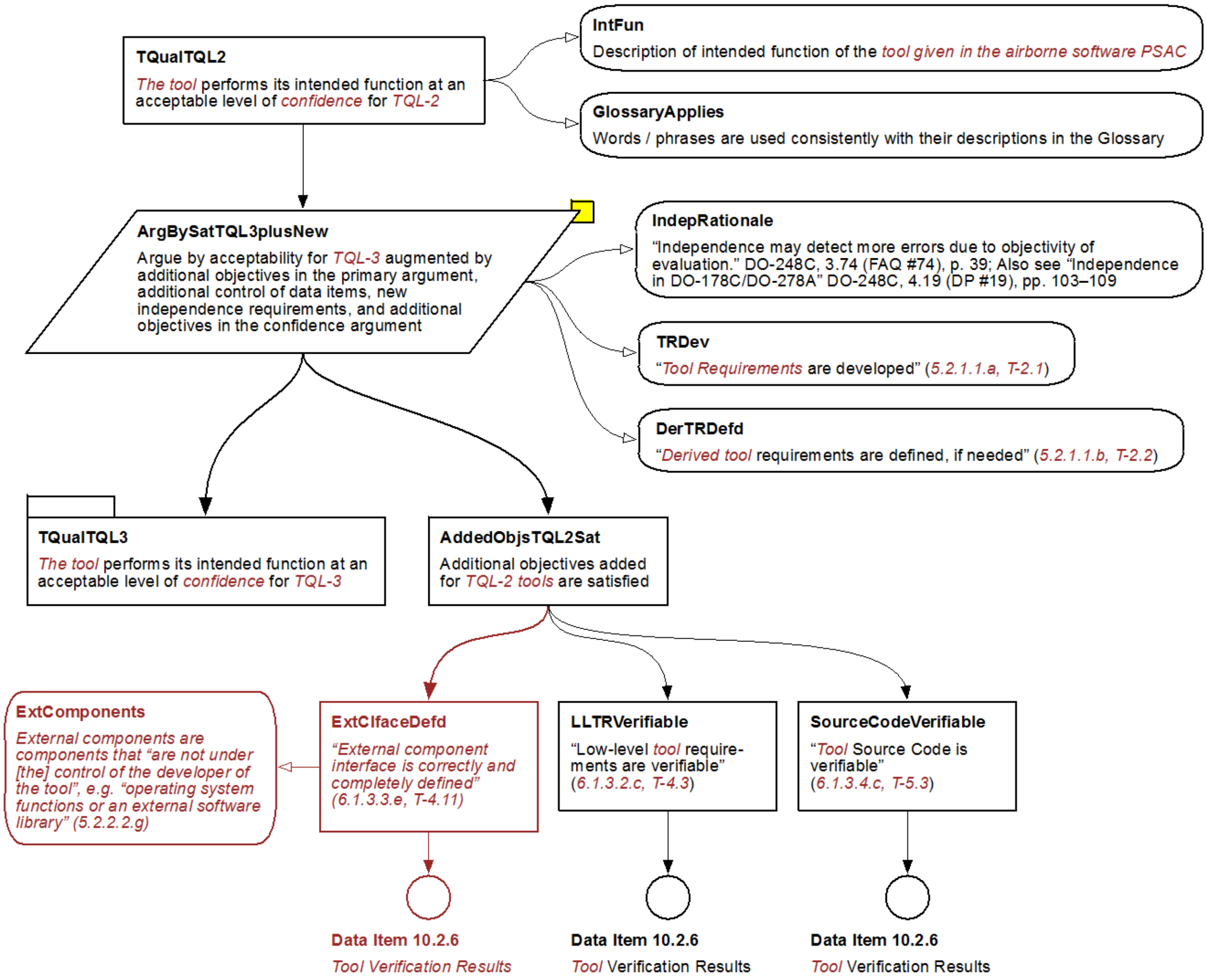

TQualTQL2
The tool performs its intended function at an acceptable level of confidence for TQL-2
IntFun
Description of intended function of the tool given in the airborne software PSAC
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
ArgBySatTQL3plusNew
Argue by acceptability for TQL-3 augmented by additional objectives in the primary argument, additional control of data items, new independence requirements, and additional objectives in the confidence argument
IndepRationale
"Independence may detect more errors due to objectivity of evaluation." DO-248C, 3.74 (FAQ #74), p. 39; Also see "Independence in DO-178C/DO-278A" DO-248C, 4.19 (DP #19), pp. 103–109
TRDev
"Tool Requirements are developed" (5.2.1.1.a, T-2.1)
DerTRDefd
"Derived tool requirements are defined, if needed" (5.2.1.1.b, T-2.2)
TQualTQL3
The tool performs its intended function at an acceptable level of confidence for TQL-3
AddedObjsTQL2Sat
Additional objectives added for TQL-2 tools are satisfied
ExtComponents
External components are components that "are not under [the] control of the developer of the tool", e.g. "operating system functions or an external software library" (5.2.2.2.g)
ExtCIfaceDefd
"External component interface is correctly and completely defined" (6.1.3.3.e, T-4.11)
LLTRVerifiable
"Low-level tool require-ments are verifiable" (6.1.3.2.c, T-4.3)
SourceCodeVerifiable
"Tool Source Code is verifiable" (6.1.3.4.c, T-5.3)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-33. TQualTQL2**

This argument extends the TQL-3 argument with three new evidence-supported premises:

1. The low-level tool requirements are verifiable (LLTRVerifiable).
2. The tool source code is verifiable (SourceCodeVerifiable).
3. The external component interface is correctly and completely defined (ExtCIfaceDefd).

These former two of these premises have analogues in the DO-178C argument supporting the conclusion that additional objectives added for Level B are satisfied (AddedObjs-LevBSat). The remaining premises in that DO-178C argument have no analogue in the DO-330 argument. That is, there is no requirement to ensure that tools are compatible with their target computers. DO-330 also does not require tool software architectures to be verifiable. The latter premise has no analogue in the DO-178C argument.

Section B.6.1 presents an argument for confidence in the TQL-2 tool-qualification argument.

### B.6.1. CONFIDENCE IN TOOL QUALIFICATION ARGUMENT FOR TQL-2

As in the DO-178C argument, confidence [demonstrating that software is correct with respect to its requirements is sufficient to show that the software will perform its intended function] would be undermined by the use of inadequate processes. Figure B-34 depicts the confidence argument for TQL-2. The argument infers that "the evidence provided is adequate for justifying that the correctness of the tool has been demonstrated to the extent needed for TQL-2" on the grounds of six premises that applied at TQL-3 and one additional premise.

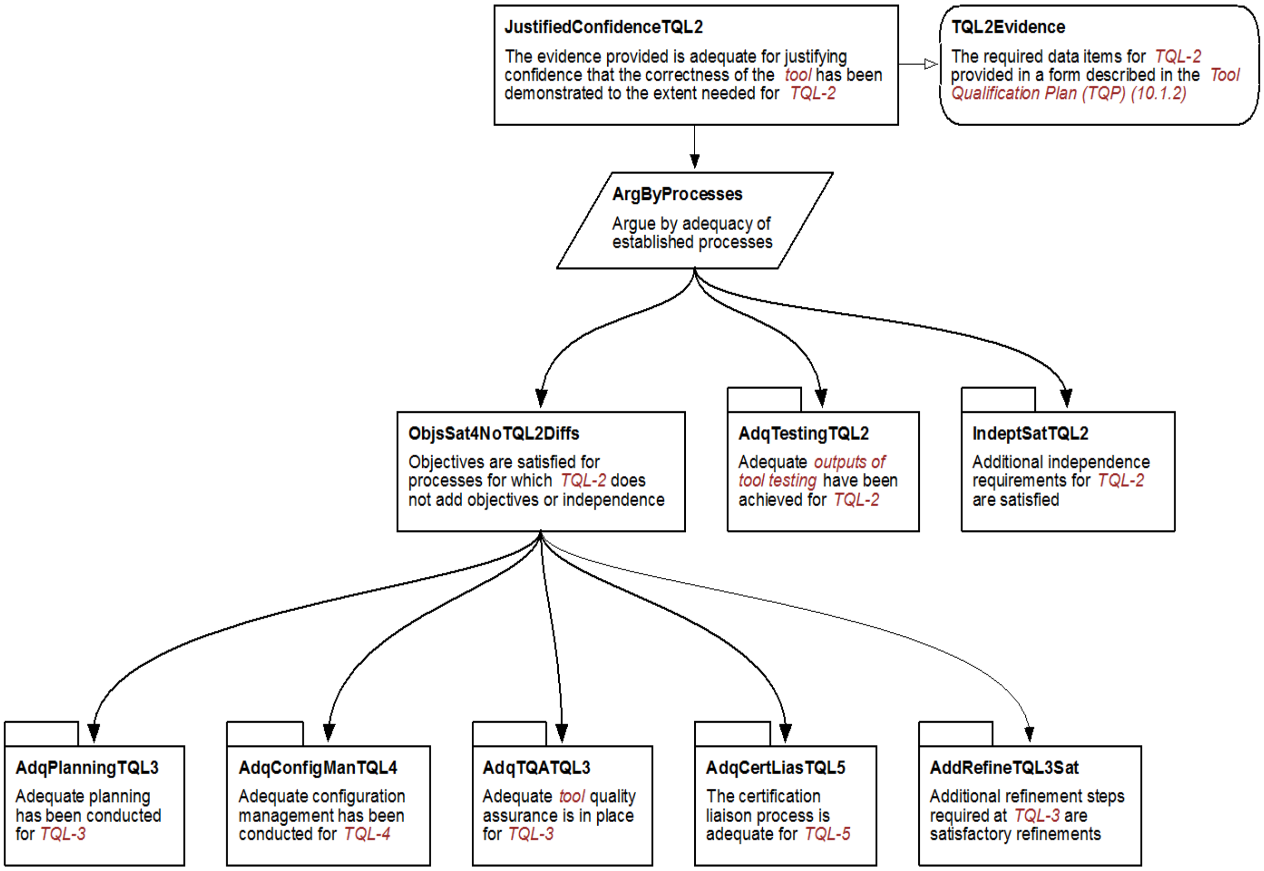

JustifiedConfidenceTQL2
The evidence provided is adequate for justifying confidence that the correctness of the tool has been demonstrated to the extent needed for TQL-2
TQL2Evidence
The required data items for TQL-2 provided in a form described in the Tool Qualification Plan (TQP) (10.1.2)
ArgByProcesses
Argue by adequacy of established processes
ObjsSat4NoTQL2Diffs
Objectives are satisfied for processes for which TQL-2 does not add objectives or independence
AdqTestingTQL2
Adequate outputs of tool testing have been achieved for TQL-2
IndeptSatTQL2
Additional independence requirements for TQL-2 are satisfied
AdqPlanningTQL3
Adequate planning has been conducted for TQL-3
AdqConfigManTQL4
Adequate configuration management has been conducted for TQL-4
AdqTQATQL3
Adequate tool quality assurance is in place for TQL-3
AdqCertLiasTQL5
The certification liaison process is adequate for TQL-5
AddRefineTQL3Sat
Additional refinement steps required at TQL-3 are satisfactory refinements

**Figure B-34. JustifiedConfidenceTQL2**

The existing premises are:

1. Adequate outputs of tool testing have been achieved (AdqTestingTQL2).
2. Adequate planning has been conducted (AdqPlanningTQL3).
3. Adequate configuration management has been conducted (AdqConfigManTQL4).
4. Adequate tool quality assurance is in place (AdqTQATQL3).
5. The certification liaison process is adequate (AdqCertLiasTQL5).
6. Additional refinement steps required at TQL-3 are satisfactory (AddRefineTQL3Sat).

The supporting argument for the latter five premises are unchanged from TQL-3. The supporting argument for the former premise, which follows, adds additional evidence to its TQL-3 analogue.

Figure B-35 presents the argument supporting the conclusion that "adequate outputs of tool testing have been achieved for TQL-2." This argument extends the analogous argument for TQL-3 (see figure B-27) and is analogous to DO-178C's Level B sufficient verification of verification results argument (AdqVerVerResLevB). The argument adds one new evidence-supported premise to its TQL-3 analogue, namely that testing achieves decision coverage (DecisionCov).

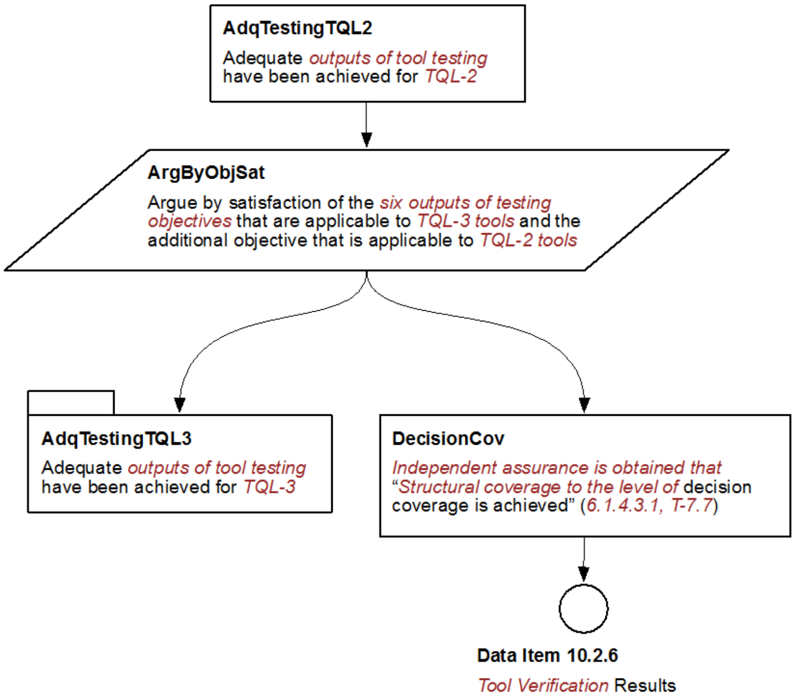

AdqTestingTQL2
Adequate outputs of tool testing have been achieved for TQL-2
ArgByObjSat
Argue by satisfaction of the six outputs of testing objectives that are applicable to TQL-3 tools and the additional objective that is applicable to TQL-2 tools
AdqTestingTQL3
Adequate outputs of tool testing have been achieved for TQL-3
DecisionCov
Independent assurance is obtained that "Structural coverage to the level of decision coverage is achieved" (6.1.4.3.1, T-7.7)
Data Item 10.2.6
Tool Verification Results

**Figure B-35. AdqTestingTQL2**

Figure B-36 depicts the argument that "additional independence requirements for TQL-2 are satisfied." This argument has no analogue at TQL-3, but is broadly similar to DO-178C's Level B additional independence argument (IndepSatLevB). The argument and its supporting arguments (shown in figures B-37–B-42) model the requirement that 16 objectives that applied at TQL-3 must be satisfied with independence at TQL-2.

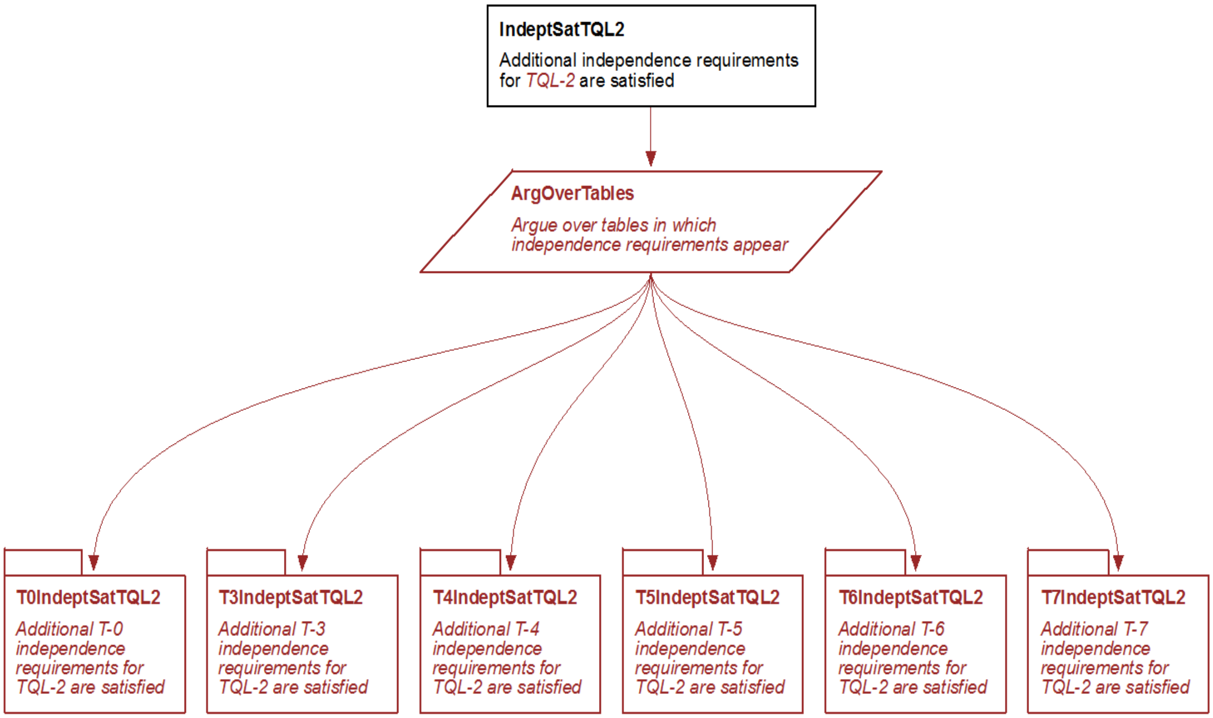

IndeptSatTQL2
Additional independence requirements for TQL-2 are satisfied
ArgOverTables
Argue over tables in which independence requirements appear
T0IndeptSatTQL2
Additional T-0 independence requirements for TQL-2 are satisfied
T3IndeptSatTQL2
Additional T-3 independence requirements for TQL-2 are satisfied
T4IndeptSatTQL2
Additional T-4 independence requirements for TQL-2 are satisfied
T5IndeptSatTQL2
Additional T-5 independence requirements for TQL-2 are satisfied
T6IndeptSatTQL2
Additional T-6 independence requirements for TQL-2 are satisfied
T7IndeptSatTQL2
Additional T-7 independence requirements for TQL-2 are satisfied

**Figure B-36. IndeptSatTQL2**

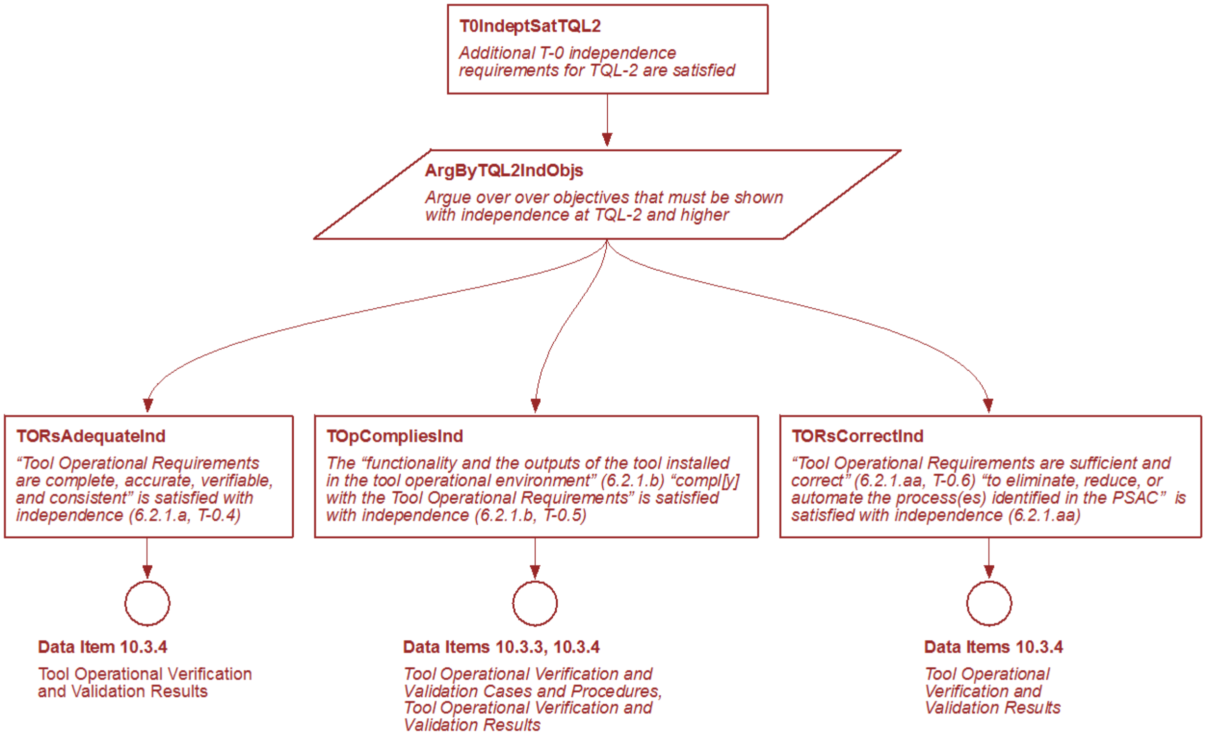

T0IndeptSatTQL2
Additional T-0 independence requirements for TQL-2 are satisfied
ArgByTQL2IndObjs
Argue over over objectives that must be shown with independence at TQL-2 and higher
TORsAdequateInd
"Tool Operational Requirements are complete, accurate, verifiable, and consistent" is satisfied with independence (6.2.1.a, T-0.4)
TOpCompliesInd
The "functionality and the outputs of the tool installed in the tool operational environment" (6.2.1.b) "compl[y] with the Tool Operational Requirements" is satisfied with independence (6.2.1.b, T-0.5)
TORsCorrectInd
"Tool Operational Requirements are sufficient and correct" (6.2.1.aa, T-0.6) "to eliminate, reduce, or automate the process(es) identified in the PSAC" is satisfied with independence (6.2.1.aa)
Data Item 10.3.4
Tool Operational Verification and Validation Results
Data Items 10.3.3, 10.3.4
Tool Operational Verification and Validation Cases and Procedures, Tool Operational Verification and Validation Results
Data Items 10.3.4
Tool Operational Verification and Validation Results

**Figure B-37. T0IndeptSatTQL2**

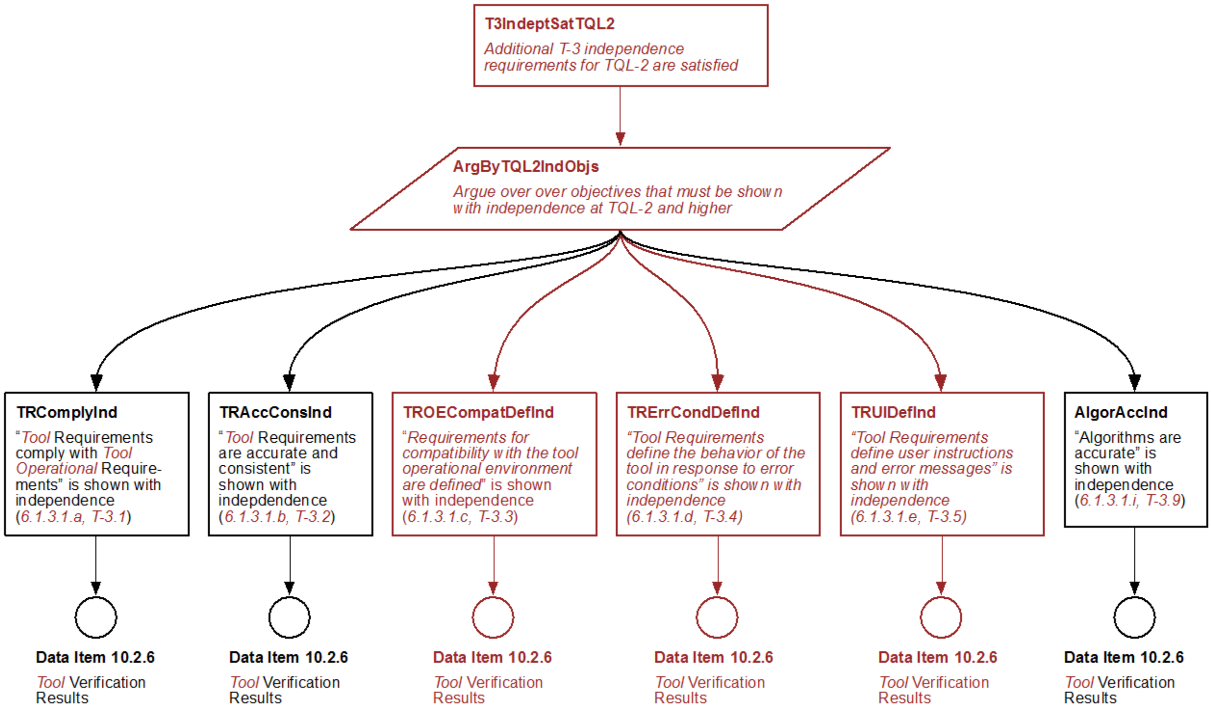

T3IndeptSatTQL2
Additional T-3 independence requirements for TQL-2 are satisfied
ArgByTQL2IndObjs
Argue over over objectives that must be shown with independence at TQL-2 and higher
TRComplyInd
"Tool Requirements comply with Tool Operational Requirements" is shown with independence (6.1.3.1.a, T-3.1)
TRAccConsInd
"Tool Requirements are accurate and consistent" is shown with indepdendence (6.1.3.1.b, T-3.2)
TROECompatDefInd
"Requirements for compatibility with the tool operational environment are defined" is shown with independence (6.1.3.1.c, T-3.3)
TRErrCondDefInd
"Tool Requirements define the behavior of the tool in response to error conditions" is shown with independence (6.1.3.1.d, T-3.4)
TRUIDefInd
"Tool Requirements define user instructions and error messages" is shown with independence (6.1.3.1.e, T-3.5)
AlgorAccInd
"Algorithms are accurate" is shown with independence (6.1.3.1.i, T-3.9)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-38. T3IndeptSatTQL2**

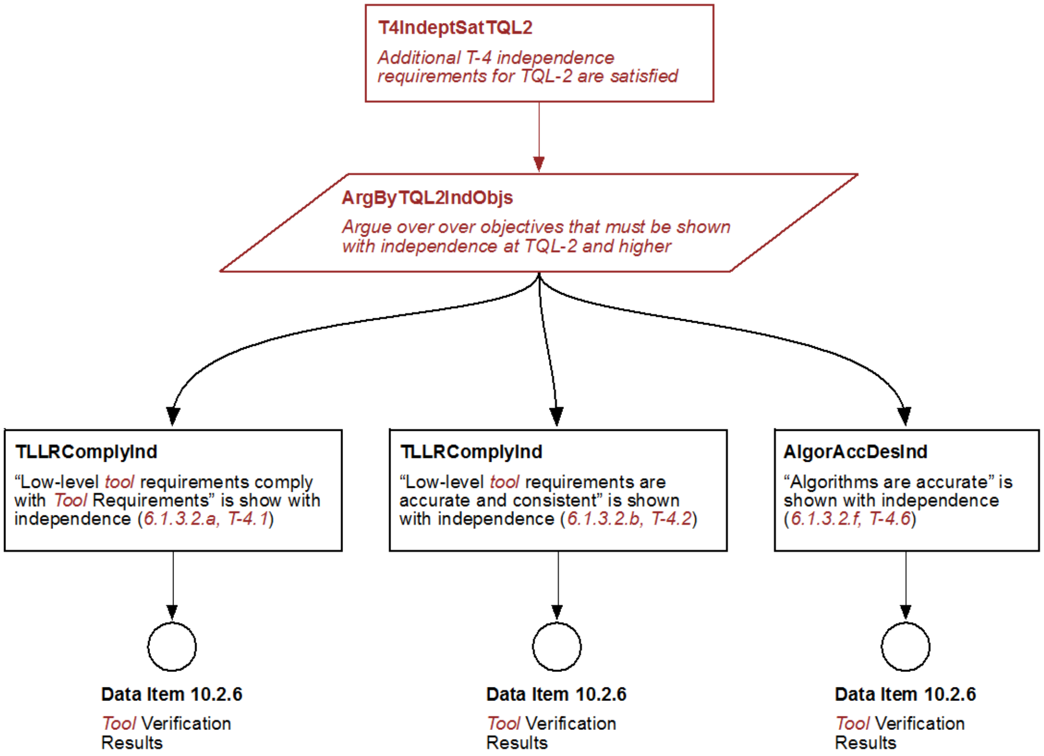

T4IndeptSatTQL2
Additional T-4 independence requirements for TQL-2 are satisfied
ArgByTQL2IndObjs
Argue over over objectives that must be shown with independence at TQL-2 and higher
TLLRComplyInd
"Low-level tool requirements comply with Tool Requirements" is show with independence (6.1.3.2.a, T-4.1)
TLLRComplyInd
"Low-level tool requirements are accurate and consistent" is shown with independence (6.1.3.2.b, T-4.2)
AlgorAccDesInd
"Algorithms are accurate" is shown with independence (6.1.3.2.f, T-4.6)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-39. T4IndeptSatTQL2**

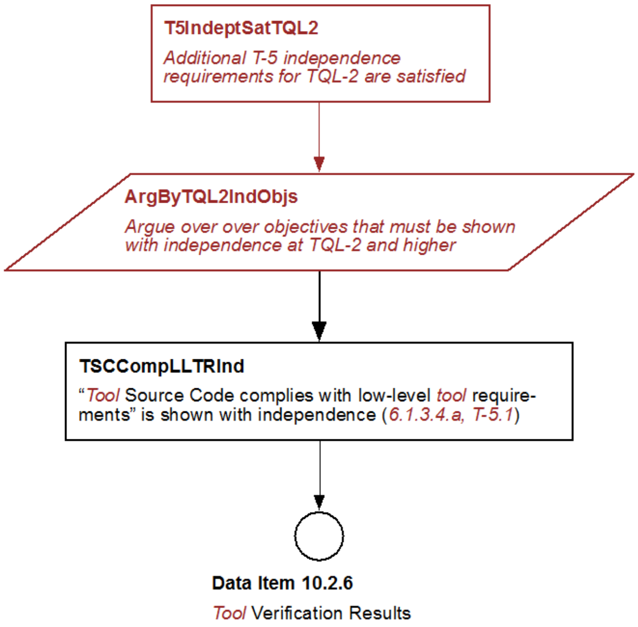

T5IndeptSatTQL2
Additional T-5 independence requirements for TQL-2 are satisfied
ArgByTQL2IndObjs
Argue over over objectives that must be shown with independence at TQL-2 and higher
TSCCompLLTRInd
"Tool Source Code complies with low-level tool requirements" is shown with independence (6.1.3.4.a, T-5.1)
Data Item 10.2.6
Tool Verification Results

**Figure B-40. T5IndeptSatTQL2**

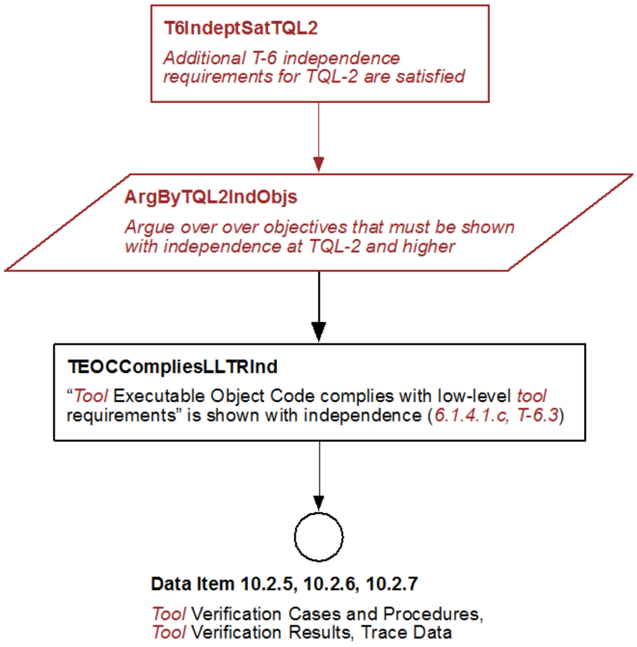

T6IndeptSatTQL2
Additional T-6 independence requirements for TQL-2 are satisfied
ArgByTQL2IndObjs
Argue over over objectives that must be shown with independence at TQL-2 and higher
TEOCCompliesLLTRInd
"Tool Executable Object Code complies with low-level tool requirements" is shown with independence (6.1.4.1.c, T-6.3)
Data Item 10.2.5, 10.2.6, 10.2.7
Tool Verification Cases and Procedures, Tool Verification Results, Trace Data

**Figure B-41. T6IndeptSatTQL2**

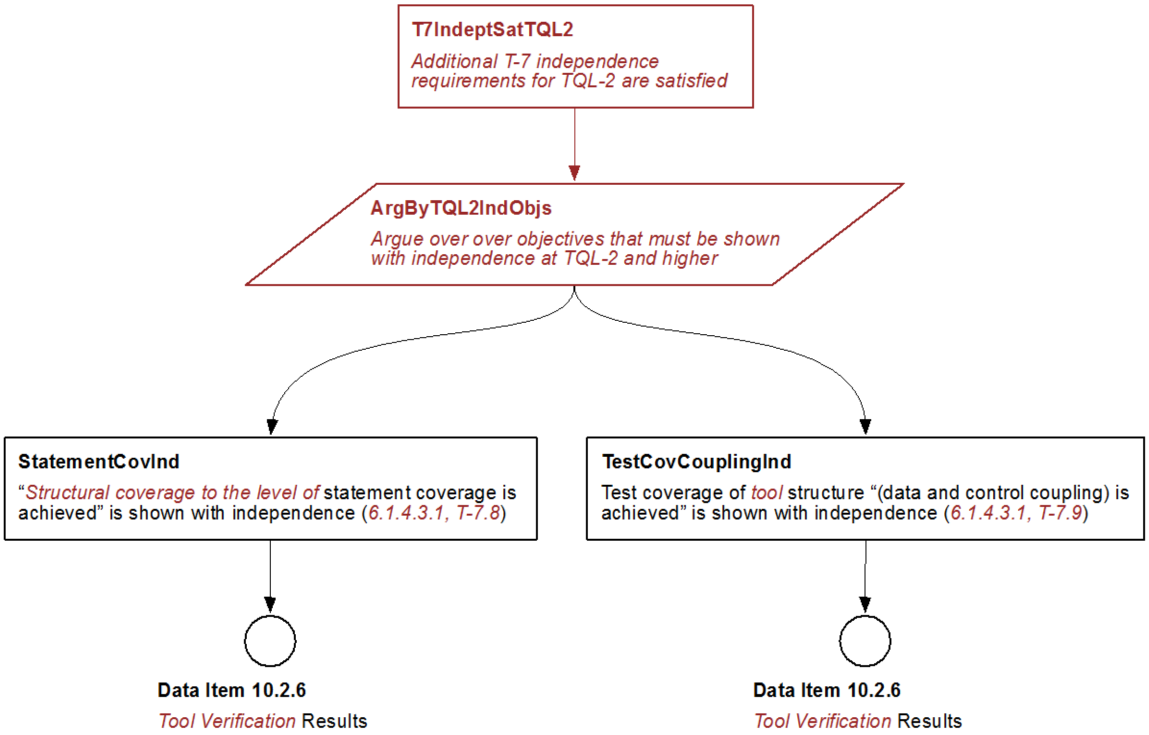

T7IndeptSatTQL2
Additional T-7 independence requirements for TQL-2 are satisfied
ArgByTQL2IndObjs
Argue over over objectives that must be shown with independence at TQL-2 and higher
StatementCovInd
"Structural coverage to the level of statement coverage is achieved" is shown with independence (6.1.4.3.1, T-7.8)
TestCovCouplingInd
Test coverage of tool structure "(data and control coupling) is achieved" is shown with independence (6.1.4.3.1, T-7.9)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-42. T7IndeptSatTQL2**

## B.7. TOOL QUALIFICATION ARGUMENT FOR TQL-1

Figure B-43 depicts the argument supporting the conclusion that a tool "performs its intended function at an acceptable level of confidence for TQL-1." This argument is similar to the tool-qualification argument for TQL-2 and the DO-178C arguments in support of the conclusion that airborne software performs its acceptable level of safety at the appropriate software level.

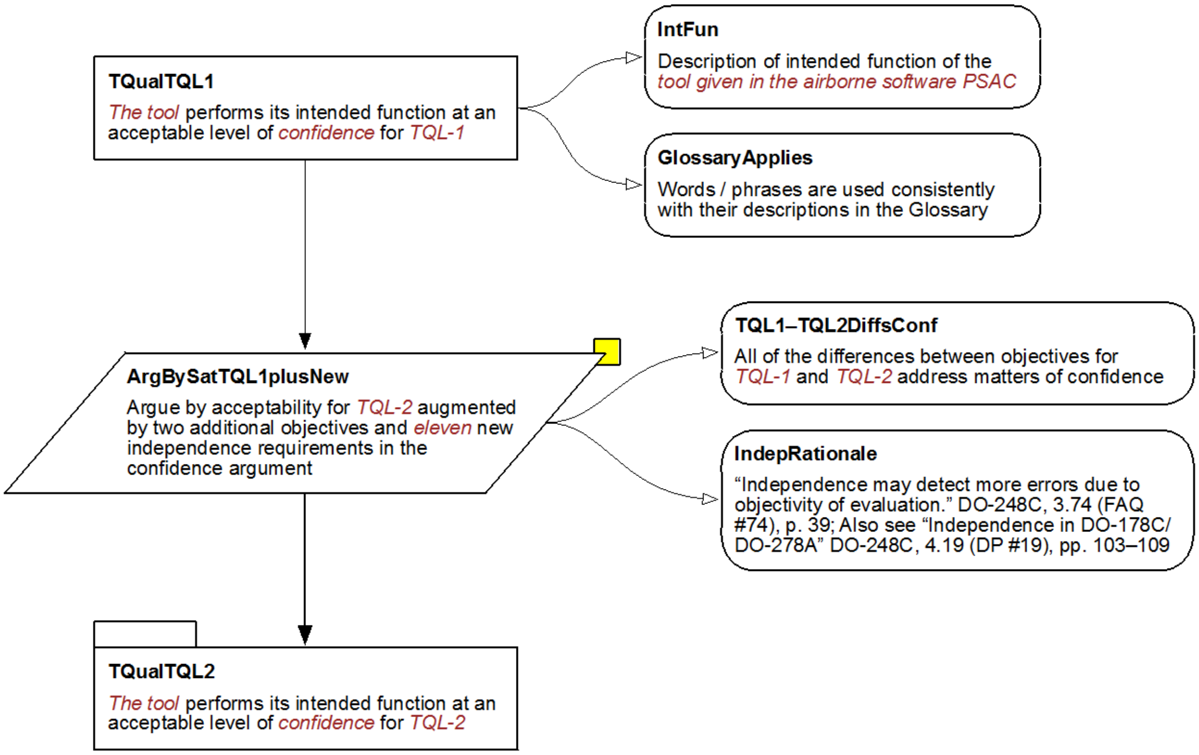

TQualTQL1
The tool performs its intended function at an acceptable level of confidence for TQL-1
IntFun
Description of intended function of the tool given in the airborne software PSAC
GlossaryApplies
Words / phrases are used consistently with their descriptions in the Glossary
ArgBySatTQL1plusNew
Argue by acceptability for TQL-2 augmented by two additional objectives and eleven new independence requirements in the confidence argument
TQL1–TQL2DiffsConf
All of the differences between objectives for TQL-1 and TQL-2 address matters of confidence
IndepRationale
"Independence may detect more errors due to objectivity of evaluation." DO-248C, 3.74 (FAQ #74), p. 39; Also see "Independence in DO-178C/ DO-278A" DO-248C, 4.19 (DP #19), pp. 103–109
TQualTQL2
The tool performs its intended function at an acceptable level of confidence for TQL-2

**Figure B-43. TQualTQL1**

This argument repeats the TQL-2 argument without adding any new premises; the differences are matters of confidence.

Section B.7.1 presents the argument for confidence in the TQL-2 tool-qualification argument.

### B.7.1. CONFIDENCE IN TOOL QUALIFICATION ARGUMENT FOR TQL-2

As in the DO-178C argument, confidence [that demonstrating that software is correct with respect to its requirements is sufficient to show that the software will perform its intended function] would be undermined by the use of inadequate processes. Figure B-44 depicts the confidence argument for TQL-1. The argument infers that "the evidence provided is adequate for justifying that the correctness of the tool has been demonstrated to the extent needed for TQL-2" on the grounds of seven premises that applied at TQL-3:

1. Adequate outputs of tool testing have been achieved (AdqTestingTQL1).
2. Additional independence requirements for TQL-1 are satisfied (IndeptSatTQL1).
3. Adequate planning has been conducted (AdqPlanningTQL3).
4. Adequate configuration management has been conducted (AdqConfigManTQL4).
5. Adequate tool-quality assurance is in place (AdqTQATQL3).
6. The certification liaison process is adequate (AdqCertLiasTQL5).
7. Additional refinement steps required at TQL-3 are satisfactory (AddRefineTQL3Sat).

The supporting argument for the latter five premises are unchanged from TQL-2. The supporting argument for the former two premises, given below, add additional evidence to their TQL-2 analogues.

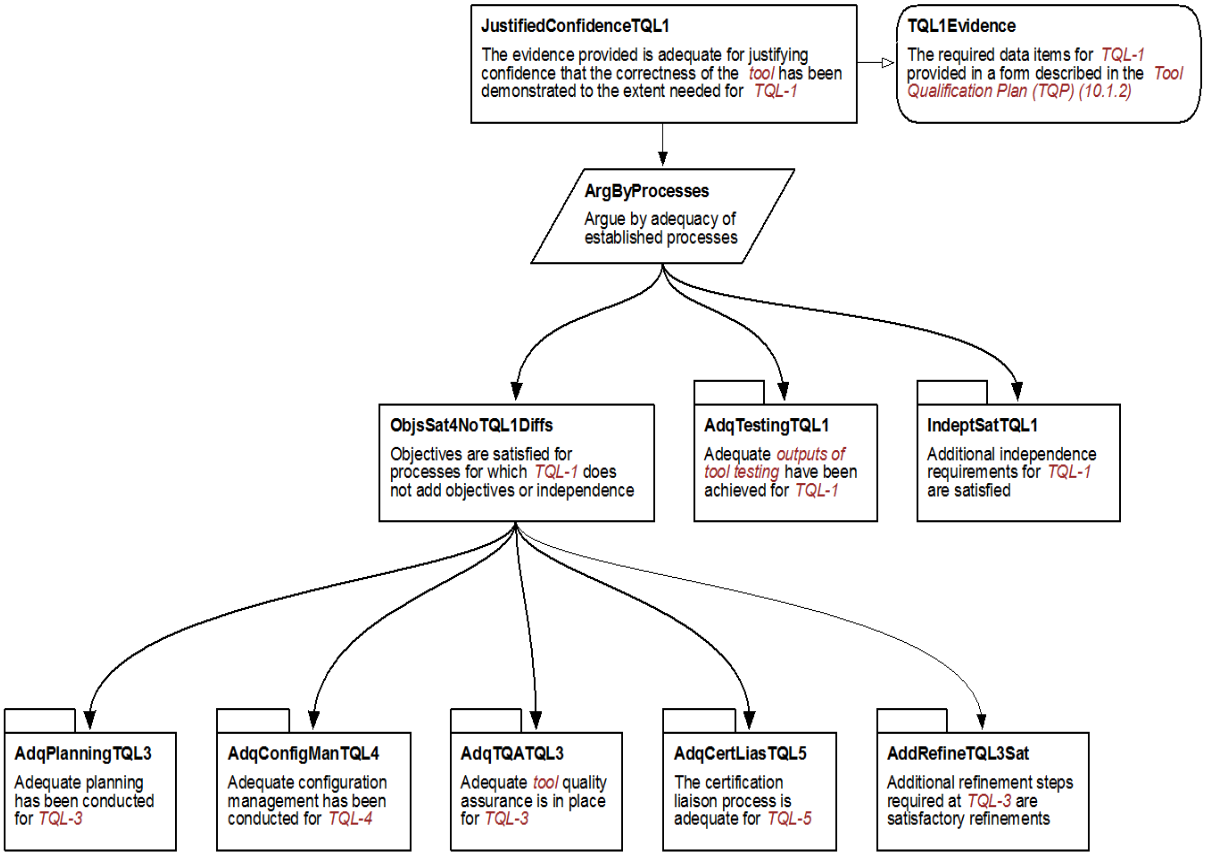

JustifiedConfidenceTQL1
The evidence provided is adequate for justifying confidence that the correctness of the tool has been demonstrated to the extent needed for TQL-1
TQL1Evidence
The required data items for TQL-1 provided in a form described in the Tool Qualification Plan (TQP) (10.1.2)
ArgByProcesses
Argue by adequacy of established processes
ObjsSat4NoTQL1Diffs
Objectives are satisfied for processes for which TQL-1 does not add objectives or independence
AdqTestingTQL1
Adequate outputs of tool testing have been achieved for TQL-1
IndeptSatTQL1
Additional independence requirements for TQL-1 are satisfied
AdqPlanningTQL3
Adequate planning has been conducted for TQL-3
AdqConfigManTQL4
Adequate configuration management has been conducted for TQL-4
AdqTQATQL3
Adequate tool quality assurance is in place for TQL-3
AdqCertLiasTQL5
The certification liaison process is adequate for TQL-5
AddRefineTQL3Sat
Additional refinement steps required at TQL-3 are satisfactory refinements

**Figure B-44. JustifiedConfidenceTQL1**

Figure B-45 gives the argument supporting the conclusion that “adequate outputs of tool testing have been achieved for TQL-1.” This argument extends the analogous argument for TQL-2 (see figure B-35) and is analogous to DO-178C’s Level A sufficient verification of verification results argument (NewVVAObjsSat). The argument adds one new evidence-supported premise to its TQL-2 analogue, namely that testing achieves modified condition/decision coverage (MCDCCov). The main difference between this argument and its DO-178C analogue is that the DO-178C argument’s premise that “verification of additional code, that cannot be trace to Source Code is achieved” has been replaced with the premise that “analysis of requirements-based testing of external components” confirms that “the tool’s code structure was verified to the degree required.”

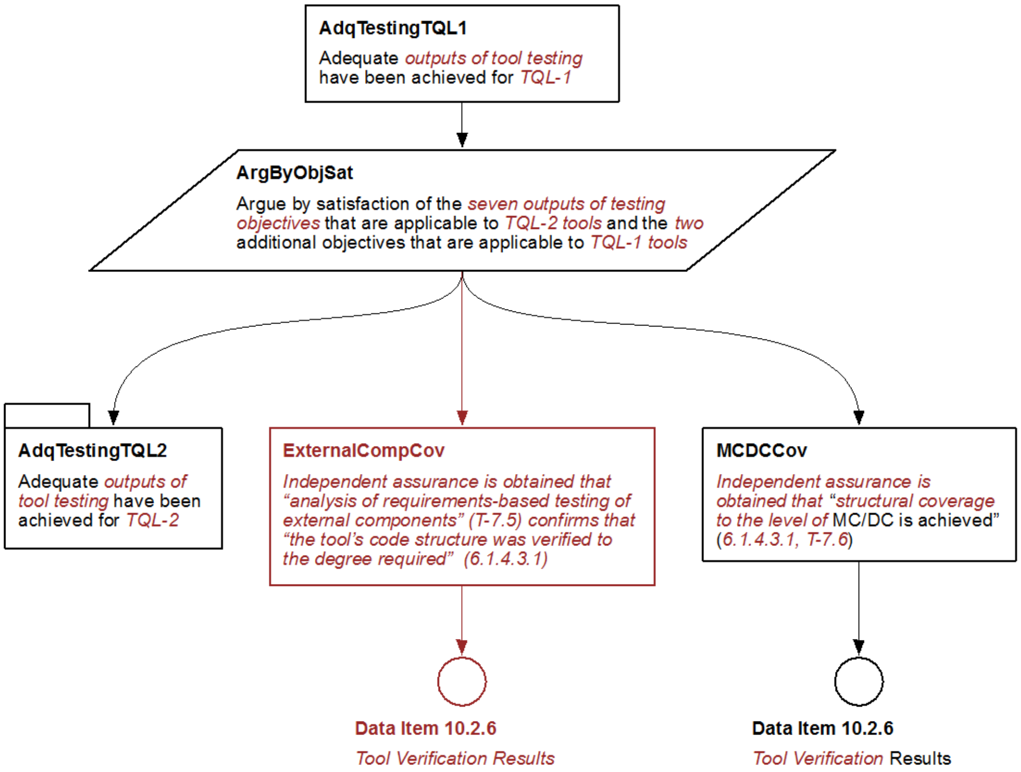

AdqTestingTQL1
Adequate outputs of tool testing have been achieved for TQL-1
ArgByObjSat
Argue by satisfaction of the seven outputs of testing objectives that are applicable to TQL-2 tools and the two additional objectives that are applicable to TQL-1 tools
AdqTestingTQL2
Adequate outputs of tool testing have been achieved for TQL-2
ExternalCompCov
Independent assurance is obtained that “analysis of requirements-based testing of external components” (T-7.5) confirms that “the tool’s code structure was verified to the degree required” (6.1.4.3.1)
MCDCCov
Independent assurance is obtained that “structural coverage to the level of MC/DC is achieved” (6.1.4.3.1, T-7.6)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-45. AdqTestingTQL1**

Figure B-46 depicts the argument that “additional independence requirements for TQL-1 are satisfied.” This argument extends an analogous TQL-2 argument (IndepSatTQL1) and is broadly similar to DO-178C’s Level A additional independence argument (IndepSatLevA). The argument and its supporting arguments (shown in figures B-47–B-50) model the requirement that 11 objectives that applied at TQL-2 must be satisfied with independence at TQL-1.

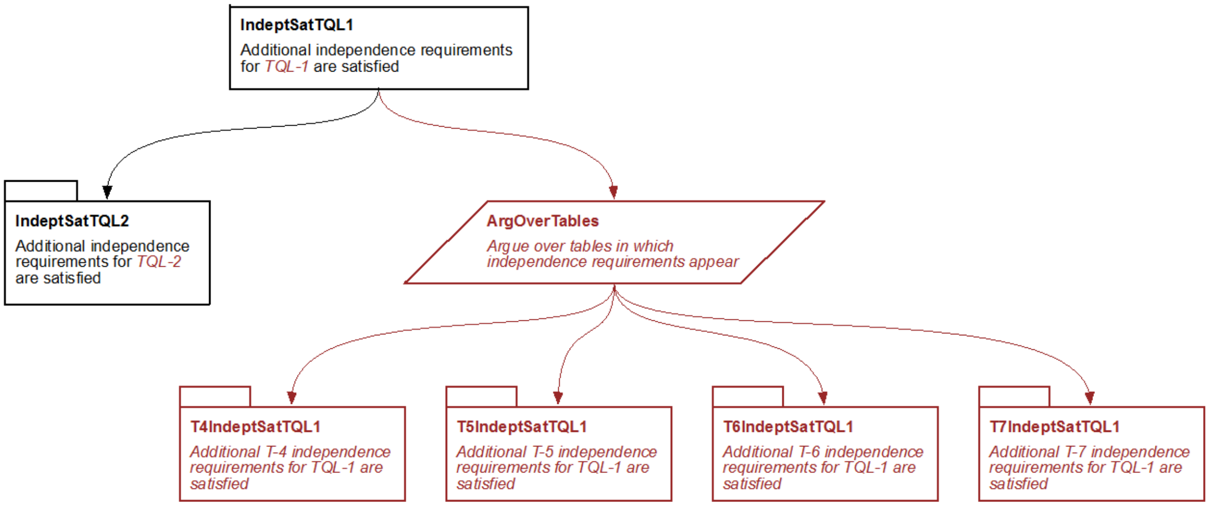

IndeptSatTQL1
Additional independence requirements for TQL-1 are satisfied
IndeptSatTQL2
Additional independence requirements for TQL-2 are satisfied
ArgOverTables
Argue over tables in which independence requirements appear
T4IndeptSatTQL1
Additional T-4 independence requirements for TQL-1 are satisfied
T5IndeptSatTQL1
Additional T-5 independence requirements for TQL-1 are satisfied
T6IndeptSatTQL1
Additional T-6 independence requirements for TQL-1 are satisfied
T7IndeptSatTQL1
Additional T-7 independence requirements for TQL-1 are satisfied

**Figure B-46. IndeptSatTQL1**

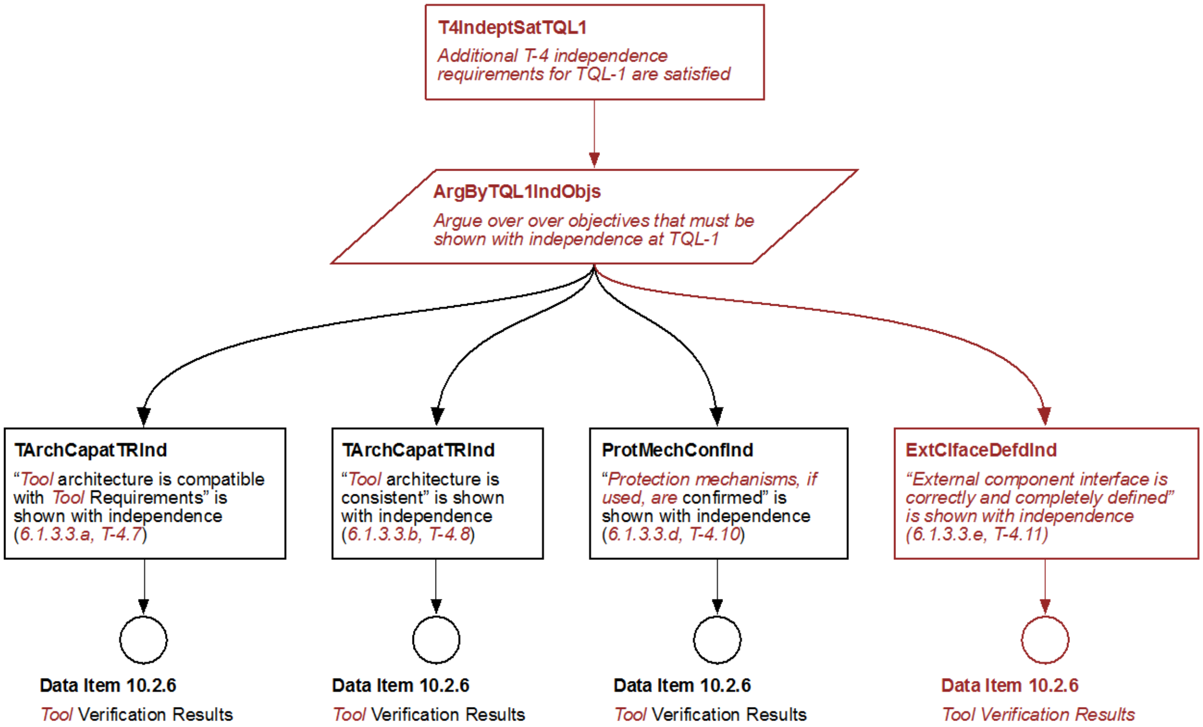

T4IndeptSatTQL1
Additional T-4 independence requirements for TQL-1 are satisfied
ArgByTQL1IndObjs
Argue over over objectives that must be shown with independence at TQL-1
TArchCapatTRInd
"Tool architecture is compatible with Tool Requirements" is shown with independence (6.1.3.3.a, T-4.7)
TArchCapatTRInd
"Tool architecture is consistent" is shown with independence (6.1.3.3.b, T-4.8)
ProtMechConfInd
"Protection mechanisms, if used, are confirmed" is shown with independence (6.1.3.3.d, T-4.10)
ExtCIfaceDefdInd
"External component interface is correctly and completely defined" is shown with independence (6.1.3.3.e, T-4.11)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-47. T4IndeptSatTQL1**

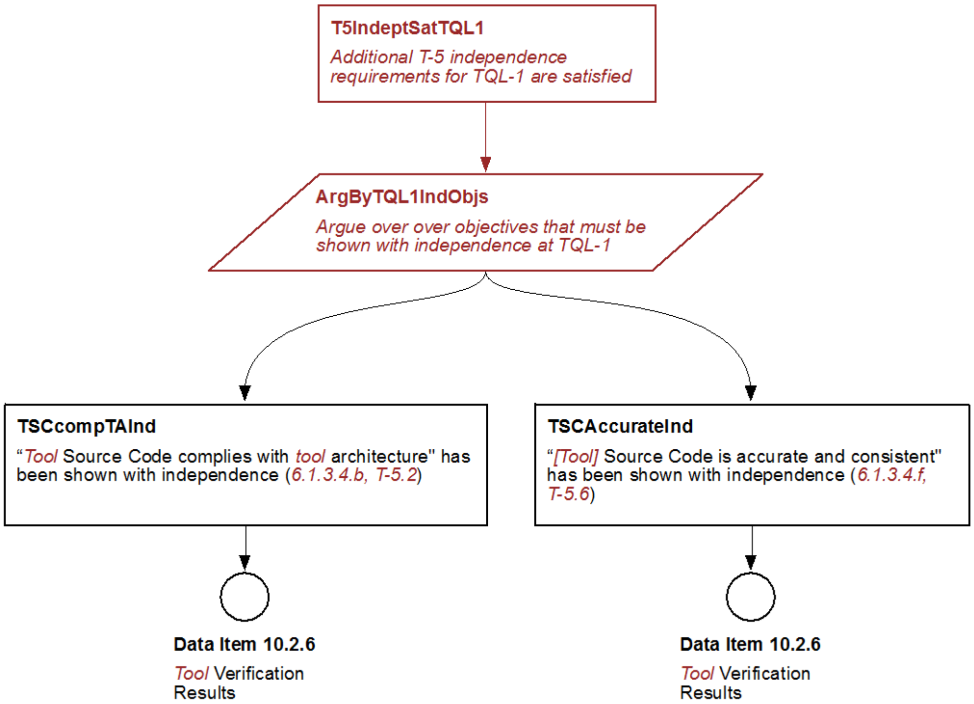

T5IndeptSatTQL1
Additional T-5 independence requirements for TQL-1 are satisfied
ArgByTQL1IndObjs
Argue over over objectives that must be shown with independence at TQL-1
TSCcompTAInd
"Tool Source Code complies with tool architecture" has been shown with independence (6.1.3.4.b, T-5.2)
TSCAccurateInd
"[Tool] Source Code is accurate and consistent" has been shown with independence (6.1.3.4.f, T-5.6)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-48. T5IndeptSatTQL1**

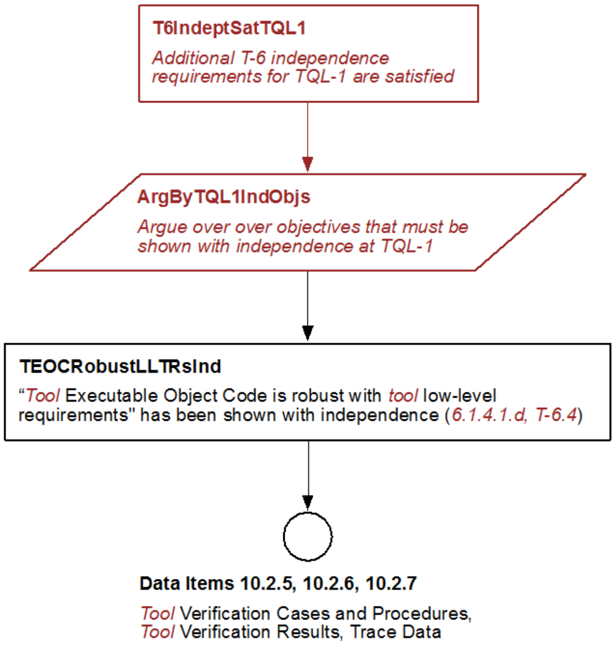

T6IndeptSatTQL1
Additional T-6 independence requirements for TQL-1 are satisfied
ArgByTQL1IndObjs
Argue over over objectives that must be shown with independence at TQL-1
TEOCRobustLLTRsInd
"Tool Executable Object Code is robust with tool low-level requirements" has been shown with independence (6.1.4.1.d, T-6.4)
Data Items 10.2.5, 10.2.6, 10.2.7
Tool Verification Cases and Procedures, Tool Verification Results, Trace Data

**Figure B-49. T6IndeptSatTQL1**

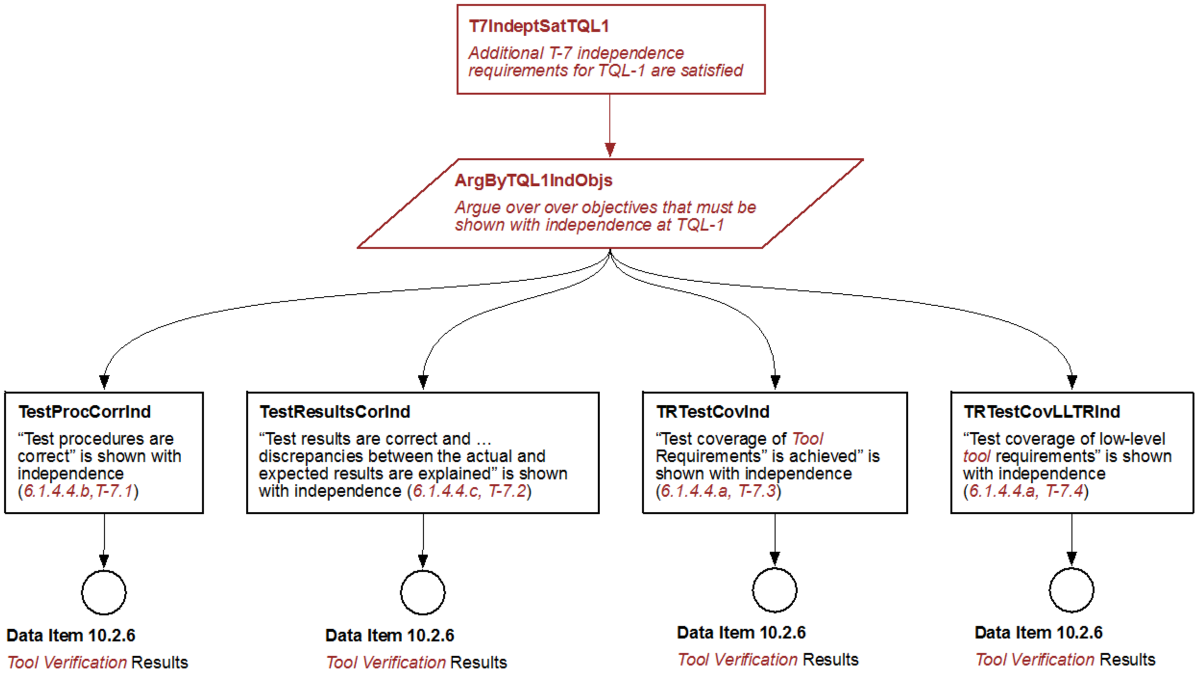

T7IndeptSatTQL1
Additional T-7 independence requirements for TQL-1 are satisfied
ArgByTQL1IndObjs
Argue over over objectives that must be shown with independence at TQL-1
TestProcCorrInd
"Test procedures are correct" is shown with independence (6.1.4.4.b,T-7.1)
TestResultsCorInd
"Test results are correct and … discrepancies between the actual and expected results are explained" is shown with independence (6.1.4.4.c, T-7.2)
TRTestCovInd
"Test coverage of Tool Requirements" is achieved" is shown with independence (6.1.4.4.a, T-7.3)
TRTestCovLLTRInd
"Test coverage of low-level tool requirements" is shown with independence (6.1.4.4.a, T-7.4)
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results
Data Item 10.2.6
Tool Verification Results

**Figure B-50. T7IndeptSatTQL1**

# APPENDIX C—PREVIOUS PAPERS

Three conference papers related to the Explicate '78 work have been published previously. These papers are included in this appendix in reverse chronological order.

- Holloway, C. M. (2015). "Explicate '78: Uncovering the Implicit Assurance Case in DO-178C." *Engineering Systems for Safety. Proceedings of the 23rd Safety-critical Systems Symposium*. M. Parsons & T. Anderson (Eds.). February 2-5. Bristol, UK. pp. 205-225. Available at <http://hdl.handle.net/2060/20150009473>. (Last accessed 27 October 2015).

- Holloway, C. M. (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. Available at <http://hdl.handle.net/2060/20140002745>. (Last accessed 27 October 2015).

- Holloway, C. M. (2012). "Towards Understanding the DO-178C / ED-12C Assurance Case." 7th IET International Conference on System Safety, Incorporating the Cyber Security Conference. October 15-18. Edinburgh, Scotland. Available at <http://hdl.handle.net/2060/20120016708>. (Last accessed 27 October 2015).

Holloway, C. M. (2015). “Explicate '78: Uncovering the Implicit Assurance Case in DO-178C.”
*Engineering Systems for Safety. Proceedings of the 23rd Safety-critical Systems Symposium.*
M. Parsons & T. Anderson (Eds.). February 2-5. Bristol, UK. pp. 205-225

# 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 exam-

[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.

Published by the Safety-Critical Systems Club.

Holloway, C. M. (2015). “Explicate '78: Uncovering the Implicit Assurance Case in DO-178C.”
*Engineering Systems for Safety. Proceedings of the 23rd Safety-critical Systems Symposium.*
M. Parsons & T. Anderson (Eds.). February 2-5. Bristol, UK. pp. 205-225

ple, 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).

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

Holloway, C. M. (2015). “Explicate '78: Uncovering the Implicit Assurance Case in DO-178C.”
*Engineering Systems for Safety. Proceedings of the 23rd Safety-critical Systems Symposium.*
M. Parsons & T. Anderson (Eds.). February 2-5. Bristol, UK. pp. 205-225

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 (bold 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**

[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.

**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 sub-claims. 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**.

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

Holloway, C. M. (2015). “Explicate '78: Uncovering the Implicit Assurance Case in DO-178C.”
*Engineering Systems for Safety. Proceedings of the 23rd Safety-critical Systems Symposium.*
M. Parsons & T. Anderson (Eds.). February 2-5. Bristol, UK. pp. 205-225

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:

1) Four fundamental concepts that greatly influence the structure and content of the e78-1.6 assurance case are discussed.
2) Salient characteristics about the case itself are provided.
3) 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).

Holloway, C. M. (2015). "Explicate '78: Uncovering the Implicit Assurance Case in DO-178C."
*Engineering Systems for Safety. Proceedings of the 23rd Safety-critical Systems Symposium.*
M. Parsons & T. Anderson (Eds.). February 2-5. Bristol, UK. pp. 205-225

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.

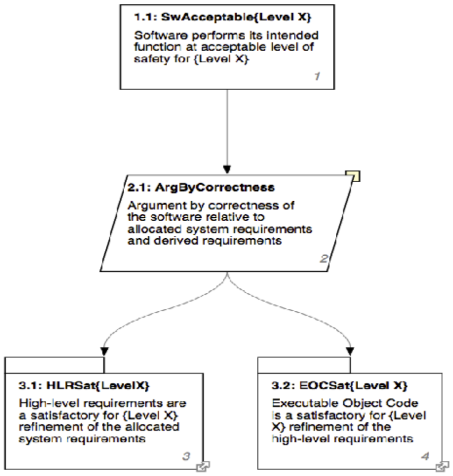

1.1: SwAcceptable{Level X}
Software performs its intended function at acceptable level of safety for {Level X}
1
2.1: ArgByCorrectness
Argument by correctness of the software relative to allocated system requirements and derived requirements
2
3.1: HLRSat{LevelX}
High-level requirements are a satisfactory 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

[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.

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.

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. Five salient points about the argument are as follows.

(1) 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.

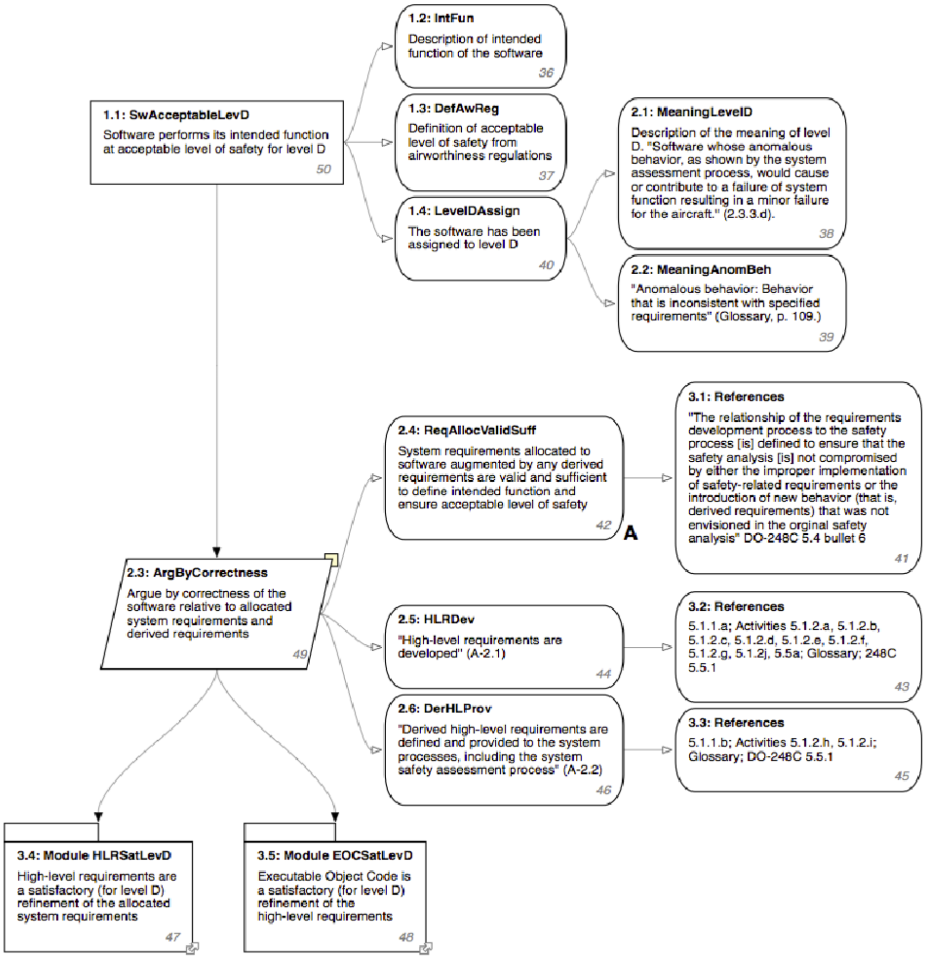

1.2: IntFun
Description of intended function of the software
36
1.1: SwAcceptableLevD
Software performs its intended function at acceptable level of safety for level D
50
1.3: DefAwReg
Definition of acceptable level of safety from airworthiness regulations
37
2.1: MeaningLevelD
Description of the meaning 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
1.4: LevelDAssign
The software has been assigned to level D
40
2.2: MeaningAnomBeh
"Anomalous behavior: Behavior that is inconsistent with specified requirements" (Glossary, p. 109.)
39
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
2.4: 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
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

(2) 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.

(3) **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.

(4) **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.

(5) 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.**

Figure 4 presents an equivalent structured text representation of the same argument.

**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.

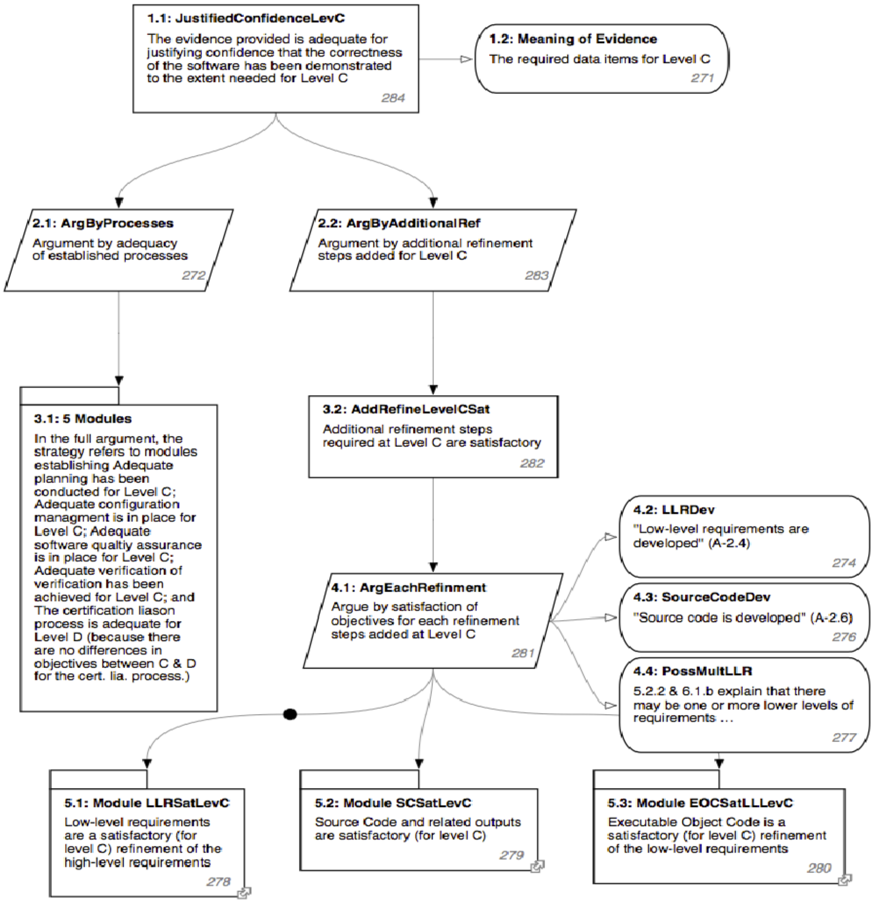

1.1: JustifiedConfidenceLevC
The evidence provided is adequate for justifying confidence that the correctness of the software has been demonstrated to the extent needed for Level C
284
1.2: Meaning of Evidence
The required data items for Level C
271
2.1: ArgByProcesses
Argument by adequacy of established 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
4.1: ArgEachRefinment
Argue by satisfaction of objectives for each refinement steps added at Level C
281
4.2: LLRDev
"Low-level requirements are developed" (A-2.4)
274
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

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.

### 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.

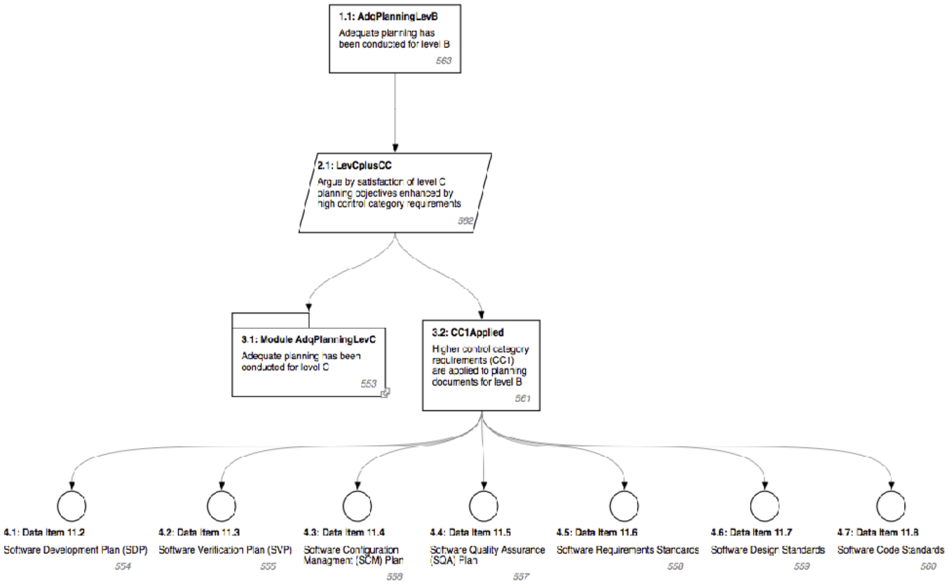

1.1: AdqPlanningLevB
Adequate planning has been conducted for level B
563
2.1: LevCplusCC
Argue by satisfaction of level C planning objectives enhanced by high control category requirements
562
3.1: Module AdqPlanningLevC
Adequate planning has been conducted for level C
553
3.2: CC1Applied
Higher control 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.

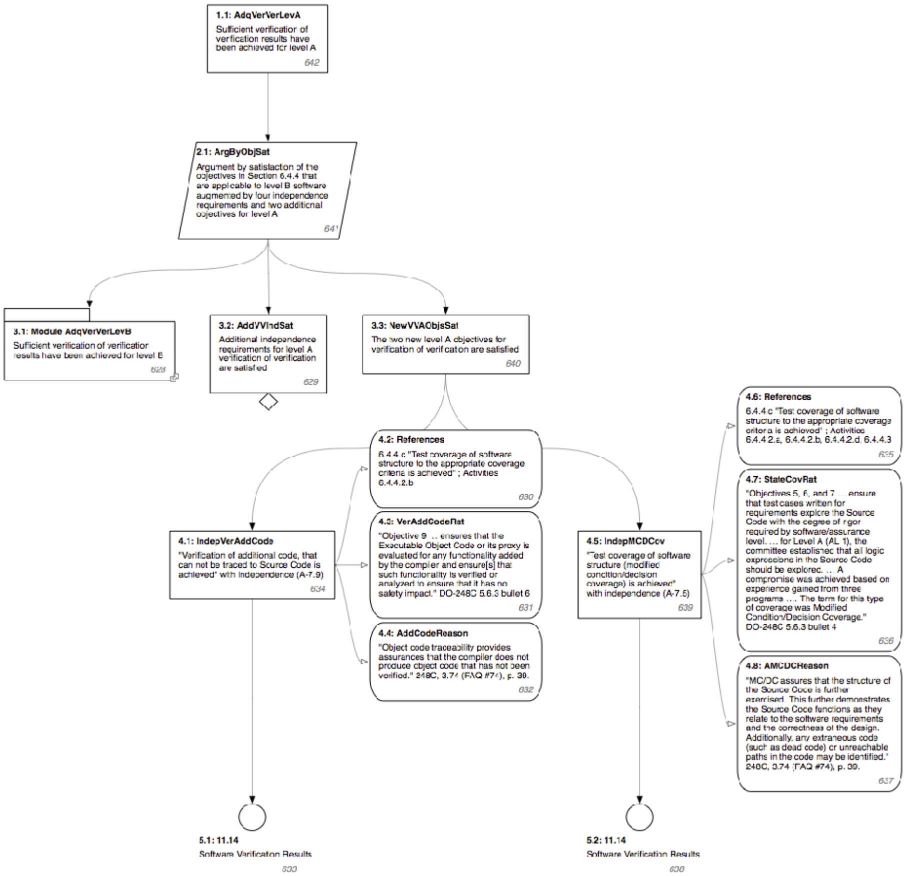

1.1: AdqVerVerLevA
Sufficient verification of verification results have been achieved for level A
642
2.1: ArgByObjSat
Argument by satisfaction of the objectives in Section 6.4.4 that are applicable to level B software augmented by four independence requirements and two additional objectives for level 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.

Holloway, C. M. (2015). “Explicate '78: Uncovering the Implicit Assurance Case in DO-178C.”
*Engineering Systems for Safety. Proceedings of the 23rd Safety-critical Systems Symposium.*
M. Parsons & T. Anderson (Eds.). February 2-5. Bristol, UK. pp. 205-225

## 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, C. M. (2015). "Explicate '78: Uncovering the Implicit Assurance Case in DO-178C."
*Engineering Systems for Safety. Proceedings of the 23rd Safety-critical Systems Symposium.*
M. Parsons & T. Anderson (Eds.). February 2-5. Bristol, UK. pp. 205-225

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)

# Making the Implicit Explicit: Towards An Assurance Case for DO-178C

C. Michael Holloway; NASA Langley Research Center; Hampton, Virginia, USA

Keywords: assurance case, translation, aviation, correctness, standards

## Abstract

For about two decades, compliance with Software Considerations in Airborne Systems and Equipment Certification (DO-178B) has been the primary means for receiving regulatory approval for using software on commercial airplanes. A new edition of the standard, DO-178C, was published in December 2011, and regulatory bodies have started the process towards recognizing this edition. The stated purpose of DO-178C remains unchanged from its predecessor: 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.” Within the text of the guidance, 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. This paper discusses a current effort to make the implicit explicit. In particular, the paper describes the current status of the research seeking to identify the specific arguments contained in, or implied by, the DO-178C guidance that implicitly justify the assumption that the document meets its stated purpose.

## Introduction

For about two decades, compliance with Software Considerations in Airborne Systems and Equipment Certification (DO-178B) (ref. 1) 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 (ref. 2).

A new edition of the standard, DO-178C, was published by the issuing bodies in late 2011 (ref. 3). New editions of two associated documents were also published at the same time: DO-278A—Software Integrity Assurance Considerations for Communication, Navigation, Surveillance and Air Traffic Management (CNS/ATM) Systems (ref. 4), and DO-248C—Supporting Information for DO-178C and DO-278A (ref. 5). Additionally four new guidance documents were published simultaneously to address specific issues and techniques: DO-330—Software Tool Qualification Considerations (ref. 6); DO-331—Model-Based Development and Verification Supplement to DO-178C and DO-278A (ref. 7); DO-332—Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A (ref. 8); and DO-333—Formal Methods Supplement to DO-178C and DO-278A (ref. 9). These seven documents have not yet received official regulatory authority approval at the time of this writing, but the regulatory bodies are well along in the process towards recognizing them[1].

The stated purpose of DO-178C remains essentially unchanged from its predecessor: 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 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. 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.

DO-178C is also mostly rationale-free, but the revised edition of DO-248C includes a new section: ‘Rationale for DO-178C / DO-278A’. This rationale section provides a basis from which building an explicit assurance case may

[1] The European Organisation for Civil Aviation Equipment (EUROCAE) uses a different document numbering scheme, but the content of the documents is otherwise identical. For example, DO-178C is identical to ED-12C. For simplicity, only the DO-numbering is referenced in this paper.

be feasible. An effort to build such a case began in September 2012, under the joint sponsorship of the Federal Aviation Administration (FAA) and the National Aeronautics and Space Administration (NASA)[2]. Preliminary work was described in (ref. 10). This paper describes the current status of the research. The remainder of the paper is organized as follows. Section 2 provides background material. Section 3 describes the process followed so far in uncovering the implicit assurance case. Section 4 provides excerpts from the current draft assurance case. Section 5 presents concluding remarks.

## Background

Fully understanding this paper requires at least a passing familiarity with DO-178B/C, the assurance case concept, and the Goal Structuring Notation (GSN) for expressing assurance cases. This section provides background information on these subjects for readers who do not already possess the requisite knowledge.

About DO-178C: Appendix A in DO-178C (ref. 3) contains a summary of the history of the DO-178 series of documents. The information in this section is derived from the appendix. The initial document in the 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 introduced the notion of five different possible software levels, with Level A denoting the highest level (for which satisfying the most rigorous objectives was required), and Level E denoting the lowest level (for which satisfying no objectives was required). The B version also introduced annex tables to summarize the required objectives by software level.

Twelve years after the adoption of DO-178B, RTCA[3] 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 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" 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 seven documents produced by the efforts—three updates and four entirely new—were enumerated in the introduction.

As a result of the terms of reference and operating instructions under which it was produced, DO-178C is 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 technology-specific supplements for formal methods, object-oriented technology, and model-based design and verification.

About assurance cases: The concept of an assurance case is a generalization of the safety case concept. 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" (ref. 11). Safety is the paramount attribute. Claims are made concerning the achievement of an acceptable level of safety, and the arguments and evidence are focused on providing justified confidence that those claims are satisfied. An assurance case, on the other hand, is concerned about providing justified confidence that claims are satisfied about additional desired attributes such as functionality, performance, or security.

Claims, arguments, evidence, context, and assumptions constitute five necessary components of a good assurance case (ref 12). *Claims* are statements about desired attributes. Other names that are used for the same concept include

[2] The joint sponsorship is under the auspices of Interagency Agreement IAI-1073: *Verification and Validation for Complex Systems*. Although the FAA has partially funded the work described in this paper, it played no part in the approval process for this paper.

[3] 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.

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 system is sufficiently safe to satisfy airworthiness requirements within its intended environment***. Examples of claims with an increasing level of specificity are as follows: ***Credible hazards have been identified***; ***Hazard H has been eliminated by design***; and ***Hardware component M has an acceptably long expected mean-time-to-failure***.

*Arguments* show how the stated claims are supported by, or justifiably inferred from, the available evidence. Other terms sometimes used for the same concept include strategies, warrants (ref. 13), and reasons. *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, properties of materials, and mathematical theorems.

*Context* 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, a description of the intended function of the software, and any constraints on the environment in which the software is expected to be used. *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 all identified hazards have been eliminated relies on the assumption **All credible hazards have been identified**.

Each of these components is present implicitly in the collective minds of the developers of any successful engineered system. An assurance case simply provides a means for ensuring that all of 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 (refs. 14-22).

About GSN: The Goal Structuring Notation is one of the most popular notations for expressing assurance cases (ref. 23). Some of the primary symbols of the notation are illustrated in figure 1. Text within these symbols is used to provide content and a convenient means of referring to individual elements.

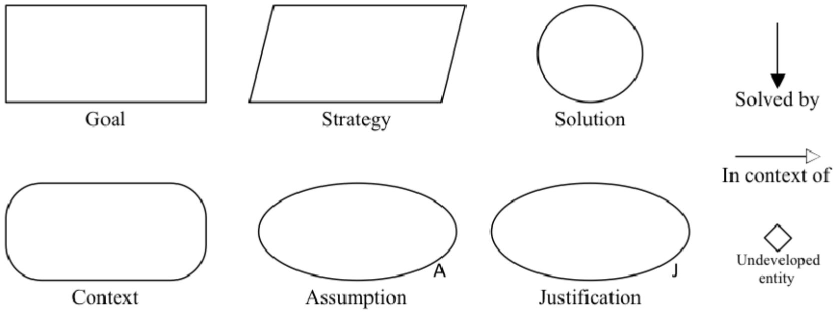


Figure 1: Main elements of GSN

The concepts represented by most of these elements have already been described. A *justification* gives the rationale for why a particular strategy or goal is acceptable. To construct an argument, the elements of the GSN notation are linked together using the *in context of* or *solved by* directed lines. The *undeveloped entity* symbol is appended to the bottom of a goal or strategy to indicate that the particular line of argument requires further development. GSN will be used in the rest of this paper to express selected portions of the preliminary assurance case that has been developed in the research. The next section explains the process that has been followed so far.

## The Process

The work to date has included the following five primary activities:

- Careful and continuing study of DO-178C and DO-248C
- Review of previous work
- Assessment of applicability of confidence argument concept
- Preliminary classification of objectives
- Selection of an approach for creating arguments

The rest of this section describes each of these activities in appropriate detail.

Study: Careful study of the text of DO-178C and relevant sections of DO-248C was the first activity in the project, and will continue throughout it. The initial study was focused on finding important context and assumptions on which the guidance rests. The results of the study were described fully in (ref. 10). For the purposes of this current paper, one item of context and one fundamental assumption are worth repeating.

The stated purpose of DO-178C, which was quoted in the introduction section of this paper, identifies a critical item of context: the airworthiness requirements. These airworthiness requirements are defined outside of DO-178C in the Code of Federal Regulations Title 14 (ref. 24). Different parts of Title 14 apply depending on the category of vehicle. Applicable categories include transport category airplanes (part 25); normal, utility, acrobatic, and commuter airplanes (part 23); transport category rotorcraft (part 29); normal category rotorcraft (part 27); products and parts (part 21); and engines (part 33). These differences mean that the top-level context for an explicit assurance case for software for a transport category airplane will be different from the context for a commuter category airplane. The former will refer to part 25, while the latter will refer to part 23.

Whereas the airworthiness context is clearly stated in DO-178C, a fundamental assumption of the guidance is discernable only through inferences from the text. This assumption involves the relationship between safety and correctness. Although in the general case, these two concepts are not equivalent (ref. 12), DO-178C rests on the assumption that within the constraints established by the guidance, establishing justifiable confidence in the correctness of the software is sufficient to establish justifiable confidence that the software does not contribute to unsafe conditions. The constraints underlying this assumption include the adequacy of the system safety processes conducted outside of the scope of DO-178C (refs. 25, 26), the effective allocation of requirements (including the requirements needed to ensure safety) to software, and the analysis by system safety processes of any new requirements that arise during software development[4].

Review: Another activity undertaken was the review of previous, related research. No published work was found that attempted to accomplish identical goals to the current effort, but two projects were uncovered that dealt with related aspects of assurance cases and DO-178B.

The MITRE Corporation conducted an effort to map three different standards into an assurance case framework (ref. 27). 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. 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, sub-goals included the following: *2.0 System Aspects are taken into account*, *4.0 Software Planning Process is executed*, *5.0 Software Development Process is executed as planned*, and *9.0 Certification Liaison process is properly established & executed.*

As best as can be determined from the published material, the effort 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.

[4] In DO-178C (and B) terminology, such requirements are called *derived requirements*. Derived requirements must be passed back to system processes, including system safety processes, for analysis of (among other things) potential safety implications.

Researchers at the University of York and QinetiQ in the United Kingdom conducted the other related previous work (ref 28). The primary goal of this research was to explore ways to justify substitution of one technology for another. In particular, the emphasis was to develop 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 sub-group responsible for creating the document that eventually become DO-333. These results have been helpful in considering various approaches to discovering and expressing a full assurance case for DO-178C.

Assess: Recent research from the University of York and the University of Virginia (ref. 18) has been even more helpful towards that end. This research introduces the idea of a *confidence argument* to accompany a primary safety argument. The safety argument documents the arguments and evidence related to direct claims of safety; the confidence argument documents the arguments and evidence 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 promise of eliminating or mitigating some of the difficulties recognized in the prevailing approach (ref. 29). Although the paper is presented in terms of a safety case, the authors acknowledge that the general concept applies equally to any property of interest.

Assessing whether the concept is appropriate for expressing the assurance argument for DO-178C was the third major activity undertaken to date. The answer was a definite yes. Even a cursory reading of the guidance reveals that it contains a mixture of objectives about the desired properties of the final software product, intermediate products, and the processes used to develop the product. A more careful reading shows that some of these objectives are naturally part of a primary argument about correctness of the final software, some are naturally part of a confidence argument that justifies appropriate belief in the sufficiency of the correctness argument, and some are a bit difficult to classify.

Classify: Conducting a preliminary classification of DO-178C objectives thus became the fourth major activity in the project. The first attempt at classification was based on the notion that every objective would likely correspond to a claim in either a correctness or confidence argument. It did not take very long to realize that this notion was too simplistic. The range of possibilities for logical correspondence of objectives not only includes claims, but also evidence, context, assumptions, and justifications. Based on this realization, the first classification was abandoned, and a second attempt was completed using the following three categories:

{1} The objective is likely to appear in some form as a claim or evidence in the primary argument.
{2} The objective is likely to appear in some form as a claim or evidence in a confidence argument.
{3} The objective is likely to appear as context, assumption, or justification in an argument (rather than as a claim or evidence).

Three examples of objectives placed in to category {1} are the following: *High-level requirements comply with system requirements* (this objective is summarized in row 1 of Table A-3, and thus often referred to as A-3.1); *Executable Object Code complies with high-level requirements* (A-6.1); and *Executable Object Code complies with high-level requirements* (A-6.5). Each of these objectives concerns properties of the final software product, and thus is directly related to a primary assurance argument. If one of these objectives is not satisfied, then it is not possible for the software to satisfy goals concerning its correctness.

Category {2} objectives include, for example, *Software plans comply with this document* (A-1.6); *Test coverage of software structure (statement coverage) is achieved* (A-7.7); and Problem reporting, change control, change review, and configuration status accounting are established (A-8.3). The reasons for the classification of these three differ. Objectives A-1.6 and A-8.3 refer to a property of the process not the product, and thus properly relate to confidence. Objective A-7-7 concerns a property of the product, but the objective does not necessarily have to be satisfied in order for the final software to satisfy goals about its correctness. It is possible for the running software to correctly implement its requirements even if the testing of the software did not cover all statements; however, higher confidence in the correctness of the software is justified if the objective is satisfied than if it is not.

Finally, the following two objectives are examples of category {3}: *The activities of the software life cycle processes are defined* (A-1.1); and *The means of compliance is proposed and agreement with the Plan for Software Aspects of Certification is obtained* (A-10.2). These objectives set part of the context within which the primary and confidence arguments reside, but are not appropriate as either claims or evidence in those arguments.

The full initial classification is summarized as follows:

- Of the 71 Level A objectives, 21 were determined to be likely to appear in some form in a primary argument, 36 in a confidence argument, and 14 as context.
- For Level B's 69 objectives, the breakdown was 20 primary, 35 confidence, and 14 context.
- For Level C's 63 objectives, the breakdown was 18 primary, 31 confidence, and 14 context.
- For Level D's 26 objectives, the breakdown was 8 primary, 10 confidence, and 8 context.

Whether these results will remain consistent throughout the remainder of the project is an open question. It seems likely that some changes will result as the primary and confidence arguments are developed and reviewed. Potential changes include not only reclassification from one category to another, but also combining multiple objectives into a single entity within an argument, and removal of some objectives from the argument entirely.

Select: Based on the results of the four activities already described, the fifth activity undertaken was determining how best to proceed in creating the initial candidate arguments. Three main questions were considered in making this determination.

Question 1 was, "What software level should be considered first?" In favor of starting with level A is the fact that the higher the level, the more important the assurance case is; thus, articulating an explicit assurance case for level A has more value than for lower levels. In favor of starting with level D is the fact that its relatively small number of objectives simplifies the tasks of discovering and articulating the explicit case, and makes reviewing the case by others easier. By increasing the likelihood of receiving constructive feedback on the initial effort, starting with level D seems likely to provide the best chance that the final product will be of high quality. So, the answer to the question was determined to be "Level D."

The second question considered was, "What notation will be used?" No single notation is ideal for everyone who may be interested in the results of the work (ref. 22); however, insufficient resources are currently available to allow expression in multiple notations. As has been already noted earlier in the paper, the answer to this question was determined to be "GSN."

"Will the developed assurance cases necessarily adhere to the DO-178C chapter / table format?" was the third main question considered. Adherence to such a format has characterized the previously published work, and dominated initial thinking in this project. Structuring sub-goals to correspond to the Annex A objectives tables seemed a natural way to proceed at first. Further reflection, however, suggested that such a structure would have two significant disadvantages: it would tend to emphasize the tables at the expense of the full text (avoiding such an emphasis was one of the goals of the C revision), and it would likely overly constrain the expression of the arguments. So, the answer to this question was determined to be "No."

With the answers determined to these questions, the initial articulation of a primary assurance argument and associated confidence arguments for Level D software could begin. The current results from that effort are presented in the next section.

## A Partial Case

The current draft primary assurance argument is presented first, divided into three parts. Afterwards a portion of one confidence argument is presented.

Primary argument: The top-level of the primary assurance argument created so far is shown in Figure 2. The overall goal *Software performs intended function at acceptable level of safety for level D* is derived from the stated purpose of DO-178C, modified for the software level. Three items of context are identified as necessary for this goal to make sense: a description of the software's intended function, the definition of software level D, and the relevant parts of the airworthiness requirements that define what constitutes an acceptable level of safety. Only the definition of software level D is provided directly in DO-178C; the others are external to the document. A critical assumption on which the entire argument rests is that the assignment of level D to the software is correct.

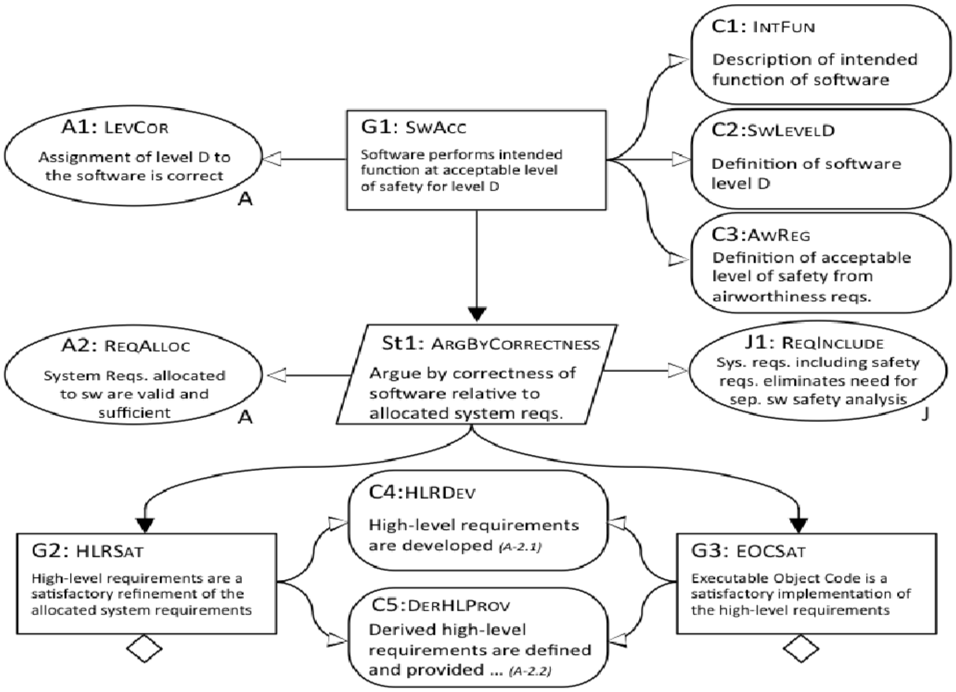

C1: IntFun
Description of intended function of software
A1: LevCor
Assignment of level D to the software is correct
A
G1: SwAcc
Software performs intended function at acceptable level of safety for level D
C2:SwLevelD
Definition of software level D
C3:AwReg
Definition of acceptable level of safety from airworthiness reqs.
A2: ReqAlloc
System Reqs. allocated to sw are valid and sufficient
A
St1: ArgByCorrectness
Argue by correctness of software relative to allocated system reqs.
J1: ReqInclude
Sys. reqs. including safety reqs. eliminates need for sep. sw safety analysis
J
C4:HLRDev
High-level requirements are developed (A-2.1)
G2: HLRSat
High-level requirements are a satisfactory refinement of the allocated system requirements
G3: EOCSat
Executable Object Code is a satisfactory implementation of the high-level requirements
C5:DerHLProv
Derived high-level requirements are defined and provided ... (A-2.2)

Figure 2: Beginning of primary argument for level D software

The DO-178C objectives and activities for Level D software imply that the implicit argument for the top-level goal G1 is based on showing that the software is correct relative to the allocated system requirements. This implicit argument relies for its cogency on (a) the assumption that the allocated system requirements are valid and sufficient with respect to the software's intended function; and (b) the justification discussed in the previous section explaining the relationship between correctness and safety in the presence of valid and sufficient requirements.

For Level D software, arguing by correctness involves two sub-goals: G2:HLRSat and G3:EOCSat. The former involves showing an appropriate relationship between the developed high-level requirements and the system requirements; the latter involves showing that the developed executable object code implements the high-level requirements. Both of these goals have meaning only within the context of high-level requirements being developed (which is DO-178C objective A-2.1), and any derived high-level requirements being provided to the system processes, including the system safety assessment processes (A-2.2). Figures 3 and 4 show further refinements of G2 and G3 respectively.

Demonstrating satisfaction of G2 comprises three sub-goals, which correspond directly to the three DO-178C objectives related to the verification of outputs of software requirements process (summarized in Table A-3) that are imposed for level D software: showing that the high-level requirements comply with system requirements (A-3.1), are accurate and consistent (A-3.2), and are traceable to system requirements (A-3.6). In the figure, context is shown only for the definition of traceable, so as to simply presentation for this paper; but the final complete assurance case will need to include context for definitions / descriptions of comply, accurate, and consistent. According to DO-178C, the evidence for satisfaction of these three objectives is contained in the Software Verification Results, which is a data item described in Chapter 11 of the guidance.

The argument for satisfaction of G3 refines into four sub-goals. Three of these sub-goals correspond to the three level D applicable objectives for testing of outputs of integration process (summarized in Table A-6); the remaining sub-goal corresponds to the only applicable objective for verification of outputs of software design process (Table A-4). All four applicable software development process (Table A-2) objectives constitute part of the relevant context for this part of the argument. One of these objectives is divided into two parts, because the portion of the objective dealing with the production of Executable Object Code (EOC) seems appropriately part of the context for G3, while the part dealing with loading of the code onto the target computer seems to be better attached to the goal concerning compatibility of EOC and target computer. The evidence for the achievement of the four sub-goals is taken from the three specific data items shown. As was the case for the G2 refinement, the refinement here for G3 shows only some of the contextual items that will need to be included in a final, complete assurance case.

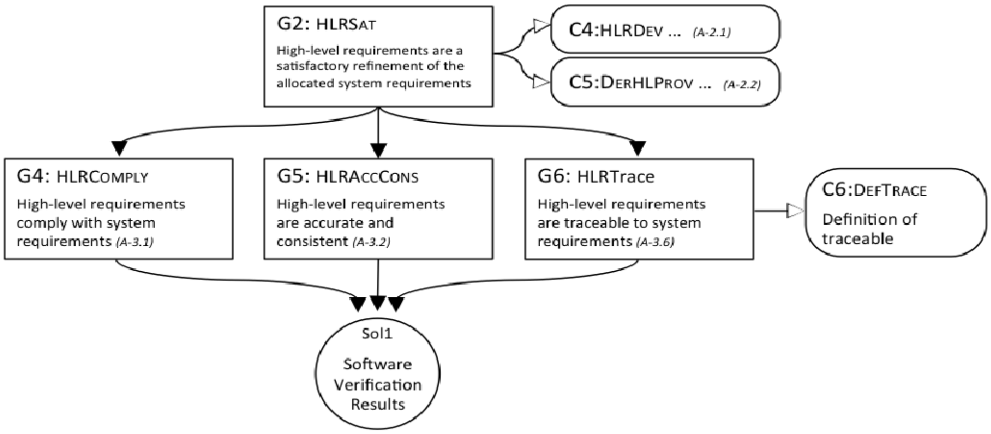

G2: HLRSAT
High-level requirements are a satisfactory refinement of the allocated system requirements
C4:HLRDEV ... (A-2.1)
C5:DERHLPROV ... (A-2.2)
G4: HLRCOMPLY
High-level requirements comply with system requirements (A-3.1)
G5: HLRACCCONS
High-level requirements are accurate and consistent (A-3.2)
G6: HLRTrace
High-level requirements are traceable to system requirements (A-3.6)
C6:DEFTRACE
Definition of traceable
Sol1
Software Verification Results

Figure 3: Refinement of G2:HLRSAT

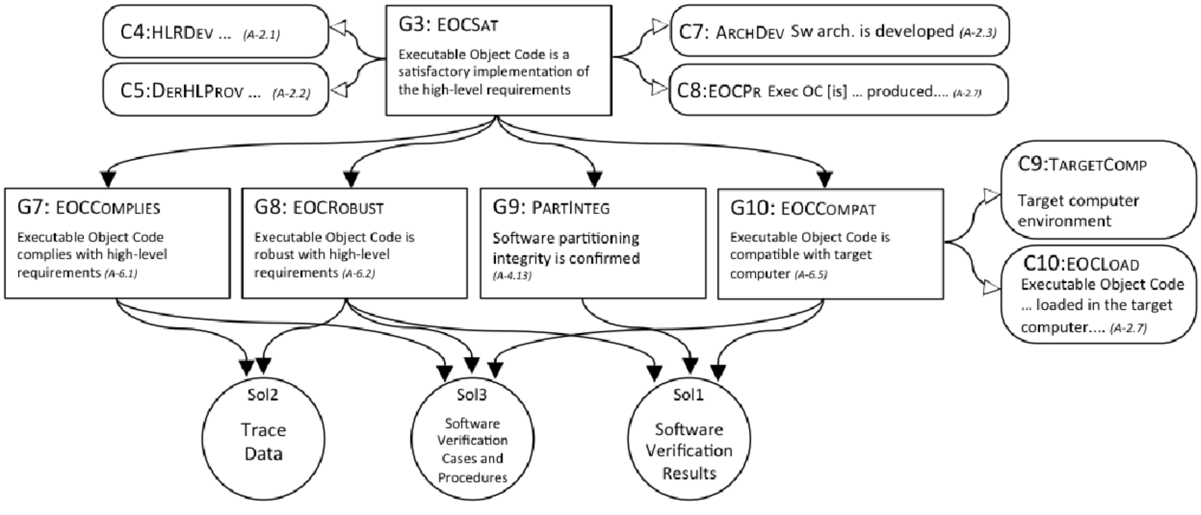

C4:HLRDEV ... (A-2.1)
C5:DERHLPROV ... (A-2.2)
G3: EOCSAT
Executable Object Code is a satisfactory implementation of the high-level requirements
C7: ARCHDEV Sw arch. is developed (A-2.3)
C8:EOCPR Exec OC [is] ... produced.... (A-2.7)
C9:TARGETCOMP
Target computer environment
C10:EOCLOAD
Executable Object Code ... loaded in the target computer.... (A-2.7)
G7: EOCCOMPLIES
Executable Object Code complies with high-level requirements (A-6.1)
G8: EOCROBUST
Executable Object Code is robust with high-level requirements (A-6.2)
G9: PARTINTEG
Software partitioning integrity is confirmed (A-4.13)
G10: EOCCOMPAT
Executable Object Code is compatible with target computer (A-6.5)
Sol2
Trace Data
Sol3
Software Verification Cases and Procedures
Sol1
Software Verification Results

Figure 4: Refinement of G3:EOCSAT

With the exception of the necessary additional contextual items already mentioned, and the absence of any goals or evidence concerning the objective related to Parameter Data Item files (A-5.8), figures 2-4 represent a complete draft articulation of the implicit primary assurance argument implied by DO-178C for level D software.

Confidence argument: Figure 5 represents a portion of one of the associated confidence arguments that have been developed so far. The goal of this argument is to illustrate the implicit reasoning in DO-178C that justifies the belief that the data items required are adequate evidence for the satisfaction of the low-level goals in the primary argument. The confidence argument relies on the adequacy of the testing of the software and of the configuration management processes in place. For level D software, the only required measure of testing adequacy is that coverage of the high-level requirements has been achieved (A-7.3); thus the testing branch of the argument has only one sub-goal.

On the other hand, DO-178C imposes on level D software the same six objectives as are required for higher software levels for configuration management; so there are six sub-goals on this branch. Three of these are shown explicitly in the figure, along with the data items that constitute the evidence for their satisfaction.

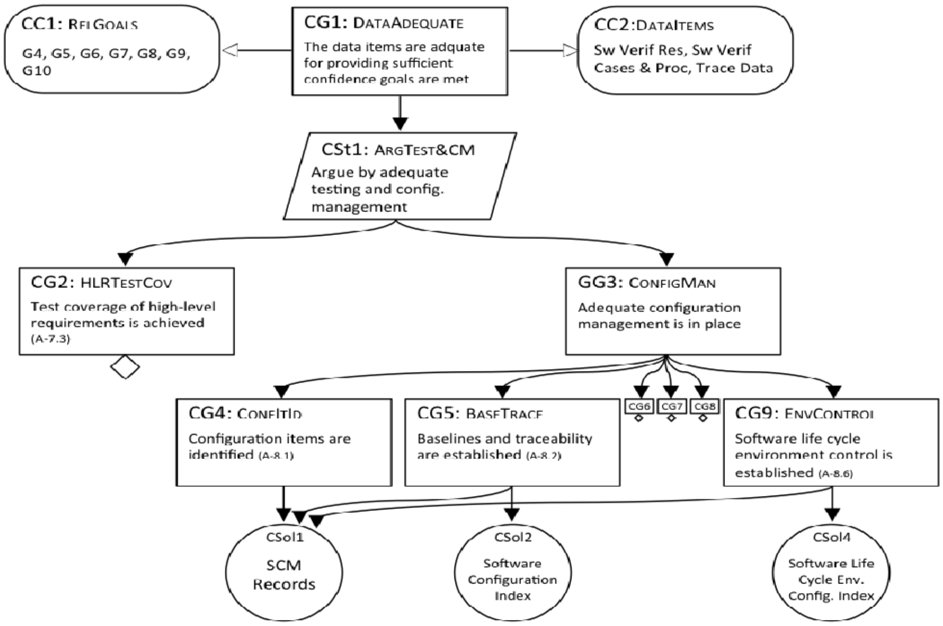

CC1: RflGoals
G4, G5, G6, G7, G8, G9, G10
CG1: DataAdequate
The data items are adquate for providing sufficient confidence goals are met
CC2:DataItems
Sw Verif Res, Sw Verif Cases & Proc, Trace Data
CSt1: ArgTest&CM
Argue by adequate testing and config. management
CG2: HLRTestCov
Test coverage of high-level requirements is achieved (A-7.3)
GG3: ConfigMan
Adequate configuration management is in place
CG4: ConfItId
Configuration items are identified (A-8.1)
CG5: BaseTrace
Baselines and traceability are established (A-8.2)
CG6
CG7
CG8
CG9: EnvControl
Software life cycle environment control is established (A-8.6)
CSol1
SCM Records
CSol2
Software Configuration Index
CSol4
Software Life Cycle Env. Config. Index

Figure 5: A partial confidence argument

Confidence arguments are also associated with (a) the reasoning from G1 through St1 to the refinement into G2 and G3; (b) the refinement of G2; and the refinement of G3. Barring unforeseen events, these confidence arguments will have been constructed by the time this paper is presented at the conference.

## Concluding remarks

This paper explained the current status of on-going research seeking to uncover the implicit assurance case that resides within the DO-178C guidance. Relevant background material was presented, completed activities were enumerated, and excerpts from the current draft articulation of an assurance case for level D software were illustrated and discussed. Comments on the draft case are welcome.

The current schedule for the research calls for completing the discovery and articulation of assurance cases for the full DO-178C guidance before the end of 2013. Assessing the utility and feasibility of conducting a similar activity for one or more of the technology supplements will begin shortly thereafter.

Three potential benefits are anticipated from successful completion of the research. First, making explicit the reasoning on which the guidance rests should enable effective analysis of the adequacy of the reasoning. Such an analysis would provide a solid foundation on which future modifications to the guidance could be based. Second, the existence of an explicit assurance case for the guidance should facilitate intelligent conversations about the relative efficacy of DO-178C and other existing or proposed approaches for demonstrating compliance with airworthiness regulations. The third potential benefit is a little bit more nebulous than the other two, but perhaps more important for the system safety community worldwide than either of them. Rightly or wrongly, DO-178 is thought by many to be a prototypical example of a prescriptive standard (refs. 12, 14). Expressing its essence as an assurance case may improve cooperation and mutual understanding between supporters of prescriptive-style standards and supporters of goal-based-style standards. At least one may hope.

## References

1. RTCA. "Software Considerations in Airborne Systems and Equipment Certification." DO-178B. 1992.

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

3. RTCA. "Software Considerations in Airborne Systems and Equipment Certification." DO-178C. 2011.

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

5. RTCA. "Supporting Information for DO-178C and DO-278A." DO-248C. 2011.

6. RTCA. "Software Tool Qualification Considerations." DO-330. 2011.

7. RTCA. "Model-Based Development and Verification Supplement to DO-178C and DO-278A." DO-331. 2011.

8. RTCA. "Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A." DO-332. 2011.

9. RTCA. "Formal Methods Supplement to DO-178C and DO-278A." DO-333. 2011.

10. Holloway, C.M. "Towards Understanding the DO-178C / ED-12C Assurance Case." *7th IET International Conference on System Safety, Incorporating the Cyber Security Conference.* Edinburgh, 2012.

11. UK Ministry of Defence. *Defence Standard 00-56 Issue 4: Safety Management Requirements for Defence Systems*. 2007.

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

13. Toulmin, S. E. *The Uses of Argument, Updated Edition*. Cambridge University Press, 2003.

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

15. Goodenough, J. B.; Weinstock, C. B.; Klein, A. Z. "Toward a Theory of Assurance Case Confidence." CMU-SEI-2002-TR-002, September 2012.

16. Graydon, P; Habli, I; Hawkins, R.; Kelly, T; Knight. J. "Arguing Conformance." *IEEE Software* 29 (3), 2012.

17. Denney, Ewen; Pai, Ganesh. "A Lightweight Methodology for Safety Case Assembly." *Proceedings of the 31st International Conference on Computer Safety, Reliability and Security (SafeComp '12)*, Magdeburg, Germany, 2012.

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

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

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

21. Hawkins, R.; Kelly, T. “A Systematic Approach for Developing Software Safety Arguments.” *Proceedings of the 27th International System Safety Conference*. Huntsville, Alabama, 2009.

22. Holloway, C. M. “Safety Case Notations: Alternatives for the Non-Graphically Inclined?” *Proceedings of the 3rd IET International System Safety Conference*. Birmingham, UK, 2008.

23. GSN Committee. Draft GSN Standard Version 1.0. <http://www.goalstructuringnotation.info/> (last accessed June 3, 2013).

24. Federal Aviation Administration. “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 June 12, 2013).

25. Society of Automotive Engineers. “Guidelines for Development of Civil Aircraft and Systems.” SAE ARP 4754a, 2010.

26. Society of Automotive Engineers. “Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.” SAE ARP 4761, 1996.

27. Ankrum, T. Scott; Kromholz, Alfred H. “Structured Assurance Cases: Three Common Standards.” Proceedings of the Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE’05). Heidelberg, Germany, 2005.

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

29. Haddon-Cave, C. *The Nimrod Review*. London: The Stationary Office, 2009. <http://www.official-documents.gov.uk/document/hc0809/hc10/1025/1025.pdf> (last accessed June 12, 2013).

## Biography

C. Michael Holloway, Senior Research Engineer, NASA Langley Research Center, 100 NASA Road, Hampton VA 23681-2199, telephone – (757) 864-1701, facsimile – (757) 864-4234, e-mail – c.m.holloway@nasa.gov.

C. Michael Holloway is a senior research computer engineer at NASA Langley Research Center. His primary professional interests are system safety and accident analysis for software-intensive systems. He is a member of the IEEE, the IEEE Computer Society, and the International System Safety Society.

Holloway, C. M. (2012). “Towards Understanding the DO-178C / ED-12C Assurance Case.” 7th IET International Conference on System Safety, Incorporating the Cyber Security Conference. October 15-18. Edinburgh, Scotland.

# 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-178C / 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).