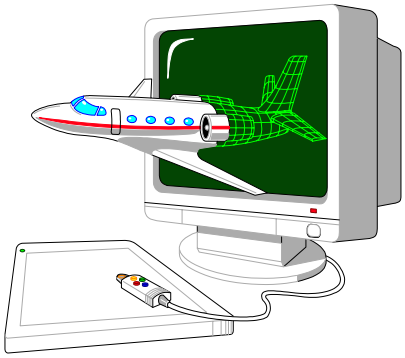
SOFTWARE REUSE IN AIRBORNE SYSTEMS

An Interactive Video Teletraining Course
IVT course # 62836
Self-Study Video #25836



Developed and Presented by:

Leanna Rierson

FAA, Chief Scientific and Technical Advisor
For Aircraft Computer Software

Aircraft Certification Service
Federal Aviation Administration

October 29-30, 2003

CONTENTS OF THIS PARTICIPANT'S GUIDE

How Do You Use This IVT Guide?

IVT Course Orientation

What Is IVT?
Who Is the Target Audience?
What Is In This Guide?
What Will You Learn?
Who Is the Instructor?
Self-Assessment (pre-course and post-course)

Appendices

A. Presentation Visuals
B. List of Acronyms
C. Exercises
D. Order 8110.49, Chapter 12
E. AC 20-RSC, Reusable Software Components
F. DO-178B Objective Considerations
G. Sample Data Sheet
H. Sample Acceptance Letter
I. Evaluation Form

How Do You Use This IVT Guide?

This Interactive Video Teletraining (IVT) Guide provides you with an orientation to the IVT presentation, support materials for use during the broadcast, and the course evaluation.

Follow these steps to complete your study:

1. Review the *IVT Presentation Orientation* before the broadcast, if possible, or before you watch the self-study videotape. It provides the purpose of the presentation, the target audience, information about the instructor, what you will learn, and topics covered.
2. Turn to Appendix A, *IVT Presentation Visuals*, and refer to it during the broadcast/videotape. You can use these visuals to take notes and follow along when viewing the presentation/self-study video.
3. Review chapter 12 of Order 8110.49, *Software Approval Guidelines* (in Appendix D), before the broadcast, if possible, or before you watch the self-study videotape.
4. Review the Advisory Circular (AC) on reusable software components (Appendix E) before the broadcast, if possible, or before you watch the self-study videotape.
5. Participate in the broadcast or watch the self-stuy videotape, performing exercises (Appendix C) and reviewing materials (Appendices D through H), as directed.
6. Complete the *IVT Presentation Evaluation Form* in Appendix I and send it to your Directorate/Division Training Manager (ATM). Your comments are very important to us and will help to enhance the quality of the IVT lesson.

NOTE: The IVT broadcast will be videotaped so that it may be used as a self-study package for those who were unable to participate in the broadcast, or for those who wish to refresh their knowledge of the content presented. This IVT Guide may also be used with the self-study videotape.

What Is IVT?

Interactive Video Teletraining, or IVT, is instruction delivered using some form of live, interactive television. This course originates from the television studio at the FAA Academy in Oklahoma City. Through the IVT broadcast facility, the instructor is able to use a variety of visuals, objects, and media formats to support the instruction.

Participants are located at various receive sites around the country and can see the instructor and his/her materials on television screens in their classrooms. The participants can communicate with the instructor either through a microphone and/or the simple-to-use Viewer Response System keypads. During the live presentation, when a participant has a question or the instructor asks for specific participant responses to questions, the participant(s) can signal to the instructor using the keypad.

The collective participant responses, or the name of a specific participant signaling a question, are immediately visible to the instructor on the console at the broadcast site. The instructor can then respond as needed. When the instructor calls on a specific participant to speak from a site, participants at each of the other sites can simultaneously hear the participant who is speaking.

Who Is the Target Audience?

Aviation safety engineers who approve airborne software.

What Is In This Guide?

This guide provides you with a framework for this course, as well as the following appendices to be used during the course:

- Appendix A contains copies of the slides used by the instructor during the broadcast. You can use these visuals to follow along with the broadcast or when you watch the tape and to record notes directly on the pages.
- Appendix B contains a list of acronyms that may be used during the broadcast. Please reference this as needed.
- Appendix C contains exercises that will be performed by students during the broadcast.
- Appendix D contains Chapter 12 of FAA Order 8110.49, *Software Approval Guidelines*. This chapter addresses the reuse of software life cycle data and will be discussed on day #2 of the broadcast.
- Appendix E contains draft 9.2 of the Advisory Circular (AC) 20-RSC entitled "*Reusable Software Components*" that will be discussed on day #2 of the broadcast.
- Appendices F through H contain information and examples that will be used to help students better understand the FAA's reuse policy and guidance.
- Appendix I contains the IVT/Self-study Evaluation Form. Please fill out this form after the IVT/self study course is finished. If taking the course "live" please fax the form to 405-954-0317. If taking this course via self-study, send the form to your Directorate/Division Training Manager (ATM) in order to get course credit.

What Will You Learn?

At the end of the training, participants will be able to:

- Explain the industry's motivation for pursuing software reuse.
- Describe technical and safety issues concerning software reuse.
- Explain FAA policy, guidance, and activities related to software reuse.
- Describe keys for safe acceptance of reused.

Who Is the Instructor?

Leanna Rierson is the FAA's Chief Scientific and Technical Advisor for Aircraft Computer Software. She has 14 years of experience in the computer/aviation industry. These positions include: national software program manager of the FAA Avionics Branch (AIR-130), avionics/electrical engineering specialist at the Wichita ACO, and software positions with industry at NCR and Cessna Aircraft Company. Leanna graduated summa cum laude from Wichita State University and has a Master's degree in Software Engineering from Rochester Institute of Technology. Leanna has led numerous national and international software teams. She has performed research in the area of software reuse for the past four years.

Self-Assessment

Pre- & Post-Course Self-Assessment Questions

If you are taking this course via IVT and you are logged on to a keypad, you will be asked before and after the broadcast to complete this self assessment, using your keypads. If you are taking this via self-study video or do not have a keypad, please complete manually and return with your end of course evaluation to your directorate/division training manager (ATM).

Rate your confidence level for each of the following statements before and after completing the course.

1. I can describe the technical concepts that make software reuse possible.

	a. Very Confident	b. Moderately Confident	c. Not Confident
BEFORE THE COURSE:	☐	☐	☐
AFTER THE COURSE:	☐	☐	☐

2. I can explain safety concerns related to software reuse.

	a. Very Confident	b. Moderately Confident	c. Not Confident
BEFORE THE COURSE:	☐	☐	☐
AFTER THE COURSE:	☐	☐	☐

3. I can describe the technical topics addressed in FAA Advisory Circular on Reusable Software Components.

	a. Very Confident	b. Moderately Confident	c. Not Confident
BEFORE THE COURSE:	☐	☐	☐
AFTER THE COURSE:	☐	☐	☐

4. I can explain Chapter 12 of Order 8110.49, which addresses reuse of software life cycle data.

	a. Very Confident	b. Moderately Confident	c. Not Confident
BEFORE THE COURSE:	☐	☐	☐
AFTER THE COURSE:	☐	☐	☐

5. I can describe keys for safe acceptance of reused software.

	a. Very Confident	b. Moderately Confident	c. Not Confident
BEFORE THE COURSE:	☐	☐	☐
AFTER THE COURSE:	☐	☐	☐

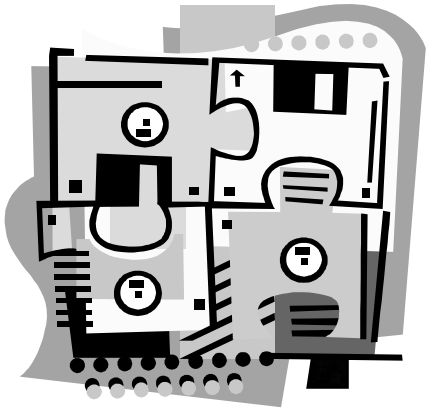
6. I can explain FAA activities related to software reuse.

	a. Very Confident	b. Moderately Confident	c. Not Confident
BEFORE THE COURSE:	☐	☐	☐
AFTER THE COURSE:	☐	☐	☐

PRESENTATION VISUALS

Appendix A

Software Reuse in Airborne Systems



Leanna Rierson
October 29-30, 2003

Ground Rules

- **Do Not Use Company Names**
- **Be Positive**
- **Be Constructive**
- **Address Controversial Topics Off-Line**

Overview of Participant's Guide

- Appendix A: Slides
- Appendix B: List of Acronyms
- Appendix C: Exercises
- Appendix D: Ch 12 of Order 8110.49
- Appendix E: AC 20-RSC (version 9.2)
- Appendix F: DO-178B Objective Considerations for RSC
- Appendix G: Sample Data Sheet

Software Reuse - **3**
October 29-30, 2003

Overview of Participant's Guide (cont)

- Appendix H: Sample Acceptance Letter
- Appendix I: Evaluation Form

Software Reuse - **4**
October 29-30, 2003

Course Objectives

- Explain the industry's motivation for pursuing software reuse.
- Describe technical and safety issues concerning software reuse.
- Explain FAA policy, guidance, and activities related to software reuse.
- Describe keys for safe acceptance of reused software.

Course Overview

Day 1:Technical Aspects of Software Reuse

- What is Reuse?
- Pros/Cons of Reuse
- Reuse Myths
- Why Reuse Isn't Used Much
- 7 Concepts Relevant to Reuse
- Successful Reuse – Pulling It All Together
- Common Certification Concerns Regarding Reuse

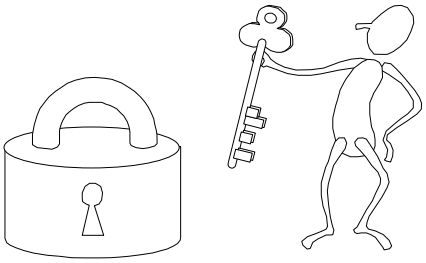
Course Overview

- Day 2: Policy & Guidance Related to Software Reuse
  - Reuse of Software Life Cycle Data (Ch 12 of Order 8110.49)
  - Reusable Software Components (AC 20-RSC)
  - Other FAA Reuse-Related Activities
  - Keys For Reuse Acceptance

Software Reuse - **7**
October 29-30, 2003

DAY 1

Technical Aspects Of Reuse



Software Reuse - **8**
October 29-30, 2003

Source: Scott Adams

GRAPHIC

What is Reuse?

What is Reuse?

- A hot buzzword?
- The newest silver bullet?
- Something greatly desired, but ever so elusive?
- Real and practical?

"If you ask five programmers what reuse is, you'll get eight answers" (Steve Adolf)

What is Reuse? (cont)

- Software Reuse != Software Salvaging (Adolf)
  - Software reuse is software that is designed to be reused
  - Software salvaging is using software that was not designed for reuse
- "Salvaging" seems to be the trend in the aviation world, but applicants want to do more "reusing"

What is Reuse? (cont)

- A process of implementing or updating software systems using existing software assets. (Sodhi)
  - Assets can be software components, objects, software requirement analysis and design models, domain architecture, database schema, code documentation, manuals, standards, test scenarios, and plans.
  - Software reuse may occur within a software system, across similar systems, or in widely different systems.

What is Reuse? (cont)

- Software reuse is the process of creating software systems from existing software assets, rather than building software systems from scratch. (Krueger)

What is Reuse? (cont)

- **Goal of reuse: To use as much software data as possible from previous development efforts in order to reduce time, cost, and risks associated with re-development.**

"Reuse is a bet on the future" (Williams)

Turn to Exercise 1 (Appendix C)

Pros/Cons of Reuse

Potential Benefits of Reuse

- **Meeting business needs (addressing the software crisis)**
- **Higher productivity**
- **Increased quality**
- **Quicker time to market**
- **Better use of resources**
- **Helps with system complexity issues**

"Systematic reuse has the highest payback of any technology since software began." (Williams)

Potential Risks of Reuse

- It requires more upfront investment
- It is a bit of a gamble on the future
- It can end up costing more, if not done properly
- It can induce errors, if not done properly
- <u>It must be used cautiously in safety-critical domains</u>

Reuse Myths

Reuse Myths

- Reuse is quick, easy, simple, & free.
- Buying components means no building.
- Components equal reuse.
- Reuse is just code.
- Maintenance is not building, therefore reuse does not apply.
- Increase productivity means loss of jobs.
- Reuse means everyone must do the same thing.

Lack of Reuse Utilization

Why Reuse Has Not Been Utilized Much

- It isn't taught in schools
- We have the "not invented here" attitude
- Cost is believed to be prohibitive
- Time constraints
- Culture
- Lack of experience
- Lack of tools
- Not understanding what reuse really is

7 Concepts Related to Reuse

7 Concepts Relevant to Reuse

1. **Planning for Reuse**
2. **Domain Engineering**
3. **Software Components**
4. **Object-Oriented Technology**
5. **Portability**
6. **Commercial-off-the-shelf (COTS) Software**
7. **Product Service History**

1. Planning for Reuse

"If you don't know where you are going, any road will lead you there." (eastern saying)

Planning for Reuse

- **Reuse doesn't just happen.**
- **Reuse must be well planned.**
- **Reuse must be well managed.**

Reifer's 10 Steps To "Reuse Adoption"

- **Define the company vision/strategy**
- **Determine company's current reuse status**
- **Establish an operation concept for the company**
- ***Prepare a business case for the company***
- **Develop a company business plan**

Reifer's 10 Steps To "Reuse Adoption" (cont)

- **Focus early efforts on company infrastructure**
- **Make an initial success**
- **Try the ideas before they are solidified**
- **Strive for a success image**
- **Iterate & refine the process based on results**

McConnell's Keys to Success in Reuse

- Take advantage of personnel continuity between old & new programs
- Do not overestimate your savings
- Secure long-term, high-level management commitment to a reuse program

McConnell's Keys to Success in Reuse (cont)

- Make reuse an integral part of the development process
- Establish a separate reuse group
- Focus on small, sharp, domain-specific components.
- Focus design efforts on abstraction & modularity.

Things To Be Addressed In Planning

- Reifer's Steps
- McConnell's Keys
- Safety
- Software/Software and Software/Hardware Integration
- Portability
- Maintenance
- Re-Verification

2. Domain Engineering

What is Domain Engineering?

- **Domain is a group or family of related systems. All systems in that domain share a set of capabilities and/or data. (Sodhi)**
- **Two Sides of Reuse:**
  - **Domain engineering → Developing <u>for</u> reuse**
  - **Reuse engineering → Developing <u>with</u> reuse**
- **Domain engineering is a developing field – it is still relatively immature**

Some Concepts of Domain Engineering

- **Knowledge reuse**
- **Repositories of components**
- **Reuse of architectural domain knowledge**
- **Reuse of software designs and patterns**
- **Reduction of "cognitive distance"**

<u>Cognitive distance</u> is the intellectual effort required to take a software system from one stage of development to another (Girardi/Ibrahim)

3. Software Components

What Is A Component?

1. *Prewritten elements of software with clear functionality and well-defined interface.* (Rhodes)

What Is A Component? (cont)

2. *An atomic software element that can be reused or used in conjunction with other components; ideally, it should work without modification and without the engineer needing to know the content and internal function of the component. However, the interface, functionality, pre-conditions, and post-conditions performance characteristics and required supporting elements must be well known.* (Lattanze)

What Is A Component? (cont)

3. A self-contained part, combination of parts, sub-assemblies or units, which performs a distinct function of a system. (DO-178B)

What Is A Component? (cont)

- AC 20-RSC → Reusable software component (RSC) is the software code and its supporting DO-178B documentation being considered for reuse. It forms a portion of the software that will be implemented by the integrator/applicant.

Examples of Components

- Real-time operating systems
- Software libraries
- Loading software
- Communication protocol stacks

A component is a piece of software and/or data that can be "chunked" by itself.

Key Properties of a Software Component

1. The component may be used by other program elements.
2. The users and developers of the software component do not need to know each other. (Meyer)

3 Attributes of Software Components

- It is reusable
- It has clear functionality
  - Single purpose
  - Encapsulates related functions
  - Properly sized
- It has well-defined interfaces
  - E.g., consistent syntax, logical design, predictable behavior, & consistent method of error handling.
  - Complete, consistent, & cohesive interfaces

8 Qualities of Software Components (Meyer)

- Careful specification of functionality & interface
- Correctness – works as specified
- Robustness – doesn't fail if used properly
- Ease of identification
- Ease of learning
- Wide-spectrum of coverage
- Consistency
- Generality – useful for multiple environments

Component Library

- Contains software components/assets to be reused throughout a company
- Library should:
  - Provide seamless access to authorized users
  - Be searchable & browsable
  - Be integrated into the engineering environment

Component Library (cont)

- **Items in the library should contain:**
  - *Design narratives* **– an overview of the component**
  - **All data that supports the component (e.g., plans, requirements, design, verification records, etc.)**
  - *Design rationale* **– detailed explanation of design decisions**

Component Library (cont)

- ***Design Rationale***
  - **Communicates the design decisions and can help users determine if it meets their needs**
  - ***Internal Design Rationale* – describes internal interaction within the component**
  - ***External Design Rationale* – describes interaction of the component with the outside world**

Component Library (cont)

- **Aspects of Libraries to be Considered:**
  - **Format of components & assets entered into the library should be useful & consistent**
  - **Best utilization of search capabilities**
  - **Library management, operation, & maintenance**

Components & Safety – Items to Consider

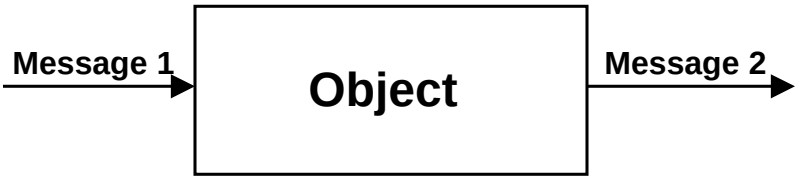
- **Planning**
- **Traceability of requirements**
- **Re-verification**
- **Interface documents**
- **Partitioning/protection**
- **Artifacts**
- **Maintenance**
- **Unused code**

4. Object-Oriented Technology (OOT)

OOT Overview

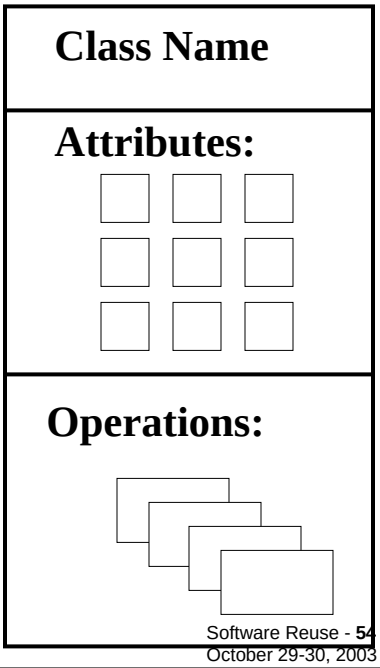
- **<u>IEEE Definition of OOT</u>: *"A software development technique in which a system or component is expressed in terms of objects and connections between those objects"***
- **Centered around "objects" and "classes"**

OOT Overview (cont)



OOT Overview (cont)

- **Definition of Class:** ***"a set of objects that share a common structure and a common behavior"*** **(Booch)**



OOT Overview (cont)

7 Major OOT Principles

Typical
- abstraction
- modularity
- concurrency
- persistence

Unique to OOT
- **encapsulation
- **hierarchy
- **typing

OOT Overview (cont)

- **Abstraction**: ***Helps to address complexity by providing crisply defined boundaries.***
- **Modularity**: ***The process of partitioning a program into logically separated and defined components that possess defined interactions and limited access to data.***
- **Concurrency**: ***Process of carrying out several events simultaneously.***
- **Persistence**: ***Property of an object through which its existence transcends time and/or space*.**

OOT Overview (cont)

- **Encapsulation:**
  - **The mechanism that binds together code and the data it manipulates**
  - **Keeps code and data safe from outside interference and misuse**
  - **Generally achieved through *information hiding***

OOT Overview (cont)

- **Hierarchy: The ordering of abstractions.**
  - **Examples of hierarchy: *single inheritance* and *multiple inheritance***
  - **Sub-class "inherits" all of the existing attributes and operations of the original class, called the "parent" or "superclass"**

OOT Overview (cont)

- Typing: Enforcement of the class of an object, such that objects of different types may not be interchanged, or at the most, they may be interchanged only in very restricted ways
  - *Polymorphism* is a concept closely related to typing.
  - *Polymorphism* comes from the Greek meaning "many forms."

OOT Methodology

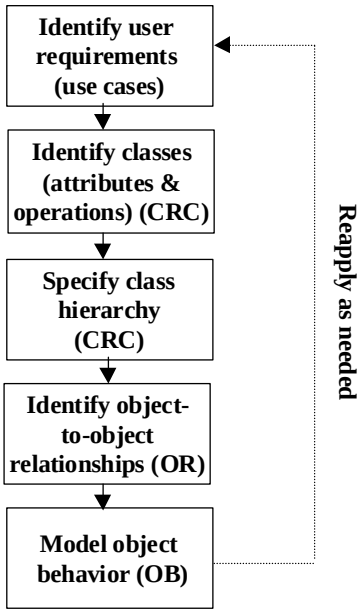
OOA = Object-Oriented Analysis

OOD = Object-Oriented Design

OOP = Object-Oriented Programming

OOV/T = Object-Oriented Testing

OOA



OOD

- **Blueprint for software construction.**
- **Four layers of design are usually defined:**
  - **subsystem layer,**
  - **class and object layer,**
  - **message layer, and**
  - **responsibilities layer.**

OOP

- Examples: SmallTalk, Java, C++, Ada 95
- C++ starting to be used in airborne avionics
- Some concerns: dynamic memory allocation, multiple inheritance, virtual base classes, run-time identification, templates, exceptions, and namespaces are deleted

OOV/T

- Process of detecting errors and verifying correctness of the OOA, OOD, and OOP. OOV/T
- Includes reviews, analyses, and tests of the software design and implementation

OOV/T (cont)

- OOV/T requires slightly different strategies and tactics than the traditional structured approach.
  - Because of inheritance, encapsulation, and polymorphism.
- Most developers use a “design for testability” approach to begin addressing any verification/test issues early in the program.

How OOT Supports Reuse

- OOT helps to break complex systems into manageable pieces
- It’s easier to implement OO design into code (using OO languages)
- OO model-based approach supports use of development tools

<u>More to come on OOT in Day 2</u>

Break

5. Portability

James Mooney
“Developing Portable Software”

Portability

- Goal of portability is transporting software to new platforms and/or environments with minimal adaptation.
- Portability is a desirable attribute for most software intended for reuse.

Note: Material based on James Mooney of University of West Virginia's work

Portability Design Strategies

- Identify the minimum necessary set of environmental requirements & assumptions.
- Eliminate all unnecessary assumptions throughout the design.

Portability Design Strategies (cont)

- **Identify specific environment interface required. For each interface, either:**
  - **Encapsulate the interface completely in a suitable module, package, object, etc; or**
  - **Identify a suitable standard for the interface, which is expected to be available in most target environments.**
- **Anticipate the need to provide a software layer to "bridge the gap" for environments which don't meet the interface assumptions.**

Technical Considerations of Portability

- **Classification of <u>Components</u>**
  - **Classify complete applications according to their environmental interfaces & requirements**
- **Specification of Portability Requirements**
  - ***how much* portability is needed**
  - ***what kind of environments* will be used**
  - ***what costs* can be accepted to achieve portability**

Technical Considerations of Portability (cont)

- **Measurement Techniques**
  - **Ways to measure portability-based cost & effectiveness**
- **Design Considerations**
  - **Portability has significant impact on the design process**
- **Cultural Adaptation**
  - **Adapting to the conventions of new environments & users**

Technical Considerations of Portability (cont)

- **Verification & Validation**
  - **Verification activities, such as reviews, analysis, & testing are needed to ensure correctness in all applications & implementations.**

Technical Considerations of Portability (cont)

- **Common Problems With Portability**
  - **OS inconsistencies**
  - **Different compiler options/effects**
  - **Incompatible libraries**
  - **Run-time problems**
  - **Underestimation of integration effort**
  - **Architectural inconsistency**

Real-Time Issues for Portability

- **Timing**
- **Memory Allocation**
- **Memory Deallocation**
- **Dynamic Task Creation**
- **Scheduling Control**
- **Synchronization & Communication**
- **Events & Input/Output**
- **File Access**
- **<u>Partitioning and Protection</u>**

6. COTS

COTS = Commercial-Off-The-Shelf

Software Reuse - **77**
October 29-30, 2003

GRAPHIC

Borrowed from Jim Krodel presentation (2002)

Software Reuse - **78**
October 29-30, 2003

COTS Software Definitions

- **RTCA/DO-178B: Commercially available applications sold by vendors through public catalog listings. COTS software is not intended to be customized or enhanced. Contract-negotiated software developed for a specific application is not COTS software.**

COTS Software Definitions (cont)

- **FAA Research Report: Any software product that is not developed within a given company for a specific application for that company. In particular, information regarding the software product's development and fabrication is not known or not available to the user of the COTS product. (Krodel)**

Two Classes of COTS

- *Class 1* – Integrity Unknown
  - No access to software life cycle data
  - Level D
  - FAA Order 8110.49 (Chapter 8)
- *Class 2* – COTS with Integrity
  - Developed using DO-178B
  - Software life cycle data exists
  - Potential use for all software levels

FAA COTS Research

- Since 1999 FAA has been sponsoring research to consider use of COTS in airborne systems.
- Research reports may be found on the software web-site: http://av-info.faa.gov/software
  - COTS HW Report (DOT/FAA/AR-01/41)
  - COTS SW Report (DOT/FAA/AR-01/26)
  - COTS RTOSs Report (DOT/FAA/AR-02/118)
  - COTS RTOSs and architectural considerations Report (SOON TO BE PUBLISHED)

FAA COTS Research (cont)

- **Phases 1 & 2: Considered both COTS hardware and software – survey of industry and potential alternatives**
  - **Revealed that the real-time operating system (RTOS) is a COTS component that many applicants plan to use.**
- **Phase 3: Considered protection techniques applicable to COTS RTOS software and developed a generic stress test plan**

FAA COTS Research (cont)

- **Phase 4: Investigated Protection Architectures/Techniques for COTS Operating Systems**
- **Research continues to look at RTOS integration approaches**

COTS Research Highlights: *Background*

- **COTS – The Hope for Reduced Development Costs**
- **Applicant's are thus applying pressures on certification authorities to approve systems with COTS Software (SW)**
- **All Airborne Software (COTS or not) Must Still Follow DO-178B**

Software Reuse - **85**
October 29-30, 2003

COTS Research Highlights: *Certification Issues*

- **COTS SW Vendor Market**
  - **Typically not aerospace – lacks DO-178B Rigor**
- **Regulatory Assessment of COTS SW**
  - **Pedigree difficult to assess**
- **Alternate methods**
  - **Reverse Engineering, Wrappers, Service History, etc. being offered for DO-178B compliance**
- **Competitive and Management Concerns**
  - **Access to records, etc.**

Software Reuse - **86**
October 29-30, 2003

COTS Research Highlights: *Other Issues*

- Vendor & Applicant Business Relationship
- Problem Reports
- Unused / Unintended Functions
- Previous COTS SW Operational Environment
- Version Control
- New Releases

Software Reuse - **87**
October 29-30, 2003

COTS Research Highlights: *COTS RTOS*

- Growing in Airborne Applications
  - Cost, Schedule Reductions, etc.
  - RTOS Services in the Aircraft Domain are Increasing
    - COTS RTOS expertise may be a better suited developer
    - Risk is lack of Vendor DO-178B knowledge
  - Several RTOS Vendors are making "DO-178B Ready RTOS' Available"

Software Reuse - **88**
October 29-30, 2003

COTS Research Highlights: *RTOS Failures With Potential Safety Impacts*

- Data consistency
- Inclusion of deactivated code or dead code
- Tasking
- Scheduling
- Memory and I/O device access
- Queuing
- Interrupts and Exceptions

COTS Research Highlights: *Robustness Test Plan (Case Study in Phase 3 & 4)*

- An approach considered to test COTS RTOSs
- Robustness Test Plan Considered:
  - Task Handling
  - Memory Management
  - Interrupt Handling

Next Steps of Research - RTOS & Component Integration

- **Partition System Development**
  - **Design Approaches & Constraints**
  - **Guidance for RTOS developers, Designers, Integrators, Applications in a Partition**
  - **Partition System Build Techniques**
  - **Building a Verifiable System**
  - **Health Monitoring Activities**
  - **Incremental Cert. Considerations**

Turn to Exercise 2 (Appendix C)

7. Product Service History

Product Service History Definition (DO-178B)

- **A contiguous period of time during which the software is operated within a known environment, and during which successive failures are recorded.**

Service History And DO-178B

- Service History is one of the alternate methods
- Acceptability for certification credit is dependent on:
  - Configuration Management of the Software
  - Effectiveness of Problem Reporting
  - Stability and Maturity of Software
  - Relevance of Product Service History Environment
  - Actual Error Rates
  - Impact of Modifications

Attributes To Be Evaluated

- Service duration length
- Change control during service
- Proposed use versus service use
- Proposed environment to service environment
- Number of significant mods during service
  - Hardware mods & software mods

Attributes To Be Evaluated (cont)

- Error detection capability
- Error reporting capability
- Number of in-service errors
- Amount/quality of service history data available and reviewed

Software Service History (SSH) Research

Although Service History seems to be a fairly straightforward technique, in practice, such use has proved extremely problematic because of the following:

- Difficulty in proving relevance of environment
- Consistency in the measure of historic performance:
  - Effectiveness of problem reporting
  - Stability/Maturity of software
  - What is the minimum "duration" of data at different criticality levels
  - How to compute "error rates"

SSH Handbook Outline

(DOT/FAA/AR-01/116 at http://av-info.faa.gov/software)

- **Introduction**
- **DO-178 Framework**
  - **The definition**
  - **Analysis of Product Service History in DO-178B (Table 1)**
  - **Relationship with Previously Developed Software**
  - **Product Service History Vs. Software Reliability**

SSH Handbook Outline

(DOT/FAA/AR-01/116) (cont)

- **The Elements of Product Service History**
  - **Questions of Problem Reporting**
  - **Questions of Operation**
  - **Questions of Environment**
  - **Questions of Time**

SSH Handbook Outline
(DOT/FAA/AR-01/116) (cont)

- **Adequacy of the Development Process**
- **Establishment of "Equivalent Safety"**
- **Summary**
- **Bibliography**
- **<u>Appendix A: Evaluation Worksheets</u>**

Analysis of PSH Guidance in DO-178B

- **Table 1 of the Handbook provides a detailed review and analysis of the eleven guidance statements for the use of product service history found in DO-178B, section 12.3.5**
- **Use Table 1 to understand "WHY" each guidance statement exists**

Analysis of PSH Guidance in DO-178B (cont)

- **Table 1 is designed to help understand the underlying rationale behind each of the guidance statements by providing:**
  - **A set of observations on what is being discussed and where some of the pitfalls may be in satisfying that guidance statement**
  - **An initial round of questions to ask regarding the available data**
  - **A clear linkage back to the elements of the definition of PSH through the Questions metaphor used throughout the Handbook**

Sample Worksheet (Table A-1)

Table A-1. Worksheet for Questions of Problem Reporting

Area:	Problem Reporting/ Configuration Management	Software Being Evaluated:	
Project:		Evaluator:	
Date:			

	DO-178B Reference	Question	Response	Issues
1.	12.3.5 a and c	Are the software versions tracked during the service history duration?		
2.	12.3.5 a and c	Are problem reports tracked with respect to particular versions of software?		
3.	12.3.5 a	Are problem reports associated with the solutions/patches and an analysis of change impact?		
4.	12.3.5 a	Is revision/change history maintained for different versions of the software?		
5.	12.3.5 a	Have change impact analyses been performed for changes?		
6.	12.3.5 b	Were in-service problems reported?		
7.	12.3.5 b	Were all reported problems recorded?		
8.	12.3.5 b	Were these problem reports stored in a repository from which they can be retrieved?		
9.	12.3.5 b	Were in-service problems thoroughly analyzed, and/or those analyses included or appropriately referenced in the problem reports?		

SSH Research Conclusions

- Worksheets- list general considerations for evaluating service history.
- Worksheets must be customized for each program.
- A list of assurance deficiencies may be derived using these worksheets for a particular program.
- Other available data as well as focused supplemental verification may be applied to complete DO-178B objectives.

SSH Research Conclusions (cont)

- Other alternate methods of compliance such as reengineering may also be applied to supplement objective evidence.
- FAA expects all of the objectives to be fulfilled regardless of what mix of methods are used to show compliance.

Other Resources on Service History

- **CAST-1 paper entitled: *"Guidance for Assessing the Software Aspects of Product Service History of Airborne Systems and Equipment" (software web-site)***
- **DO-248B discussion paper #4 (section 4.4) entitled: *"Service History Use – Rationale for DO-178B/ED-12B, Section 12.3.5a through k"***

Turn to Exercise 3 (Appendix C)

Break

Successful Reuse - Pulling It All Together

Characteristics of Organizations with Highest Reuse

- Use a product-line approach
- Utilize an architecture which standardizes interfaces and data formats
- Use common software architecture across product lines
- Implement a design for manufacturing approach

Characteristics of Organizations with Highest Reuse (cont)

- Use domain engineering
- Have a defined software reuse process
- Management understands reuse issues.
- Have software reuse advocate(s) in senior management

Characteristics of Organizations with Highest Reuse (cont)

- Employ state-of-the-art reuse tools and methods
- Reuse more than just code (e.g., requirements and design)
- Trace end-user requirements to the components which support them

Rine/Sonnerman

Golden Rule of Reuse

- Before you can reuse something, you need to:
  - Find it
  - Know what it does
  - Know how to reuse it

~ Tracz ~

REBOOT – Reuse Maturity Model (RMM)

- REBOOT = REuse Based on Object Oriented Techniques
- Implements 5 Levels Like the SEI Capability Maturity Model

REBOOT – Reuse Maturity Model (RMM) (cont)

- Level 1 – Initial or Chaotic
  - No planned reuse
  - Only unintentional reuse occurring
- Level 2 – Repeatable
  - Project-to-project reuse
  - Limited scope
  - No overall reuse strategy

REBOOT – Reuse Maturity Model (RMM) (cont)

- **Level 3 – Defined**
  - **Defined company-wide reuse strategy**
  - **Defined processes allow for reuse across the company**
  - **Company-wide reuse library**
  - **Each project is evaluated for reuse potential in accordance with the company's reuse strategy**

REBOOT – Reuse Maturity Model (RMM) (cont)

- **Level 4 – Managed**
  - **Reuse processes and reusable assets of the company are controlled and understood in detail.**
- **Level 5 – Optimized**
  - **Quantitative feedback**
  - **Continuously improve reuse processes & assets**
  - **Innovative ideas are evaluated and applied**

REBOOT – Reuse Maturity Model (RMM) (cont)

- **Key Reuse Areas**
  1. **Reuse commitment**
  2. **Project management**
  3. **Asset management**
  4. **Metrics**
  5. **Development process**

Certification concerns

Common Certification Concerns With Reuse

- Lack of insight into the software being reused (e.g., COTS)
- Lack of data to support the certification effort
- Failure to PLAN for reuse in the original development of the software (i.e., Salvaging vs. reuse)
- Dead or deactivated code that may exist in the reused software

Common Certification Concerns With Reuse (cont)

- Robustness of the reused software in a new application
- Interface challenges
- Data and control coupling
- In-service problems with the software to be reused
- Requirements levels and definitions
- Traceability between the reused software and its new application

Common Certification Concerns With Reuse (cont)

- Determining the amount of re-verification needed
- Trust that the first usage was adequately assured
- Determining the suitability of the new domain

Turn to Exercise 4 (Appendix C)

Summary of Day 1

Summary of Day 1

- **Reuse is desirable for many reasons**
- **We explored seven concepts related to reuse**
- **Reuse requires careful planning**
- **There are a number of certification and safety concerns regarding software reuse**
- **Safety must be a priority**
- **FAA has several initiatives underway to help address reuse**

What's Coming on Day 2

- **FAA Policy, Guidance, & Activities Related to Software Reuse**
  - **Reuse of Software Life Cycle Data (Ch 12 of Order 8110.49)**
  - **Reusable Software Components (AC 20-RSC)**
  - **Other FAA Reuse-Related Activities**
  - **Keys Reuse Considerations**

DAY 2

Policy, Guidance, & Activities Related to Reuse

Summary of Day 1

Course Overview

- <u>Day 1</u>:Technical Aspects of Software Reuse
  - What is Reuse?
  - Pros/Cons of Reuse
  - Reuse Myths
  - Why Reuse Isn't Used Much
  - 7 Concepts Relevant to Reuse
  - Successful Reuse – Pulling It All Together
  - Common Certification Concerns Regarding Reuse

Overview of Day 2

FAA Reuse-Related Activities

- **Reuse of Software Life Cycle Data (Ch 12 of Order 8110.49)**
- **Reusable Software Components (AC 20-RSC)**
  - **Why the AC is needed**
  - **Common misconceptions regarding AC 20-RSC**
  - **Overview of AC 20-RSC**
  - **Experience using the AC 20-RSC concept**

FAA Reuse-Related Activities (cont)

- **Other FAA Reuse-Related Activities**
  - **COTS Research**
  - **OO Handbook and Research**
  - **Software Service History Handbook**
  - **Integrated Modular Avionics**
  - **Tool Research and Reuse**
- **Keys For Accepting Reused Software**

Order 8110.49 (Chapter 12)

Title of Order: **Software Approval Guidelines**
Title of Chapter 12: **Approving Reused Software Life Cycle Data**
Date of Release: **June 3, 2003**

SEE APPENDIX D

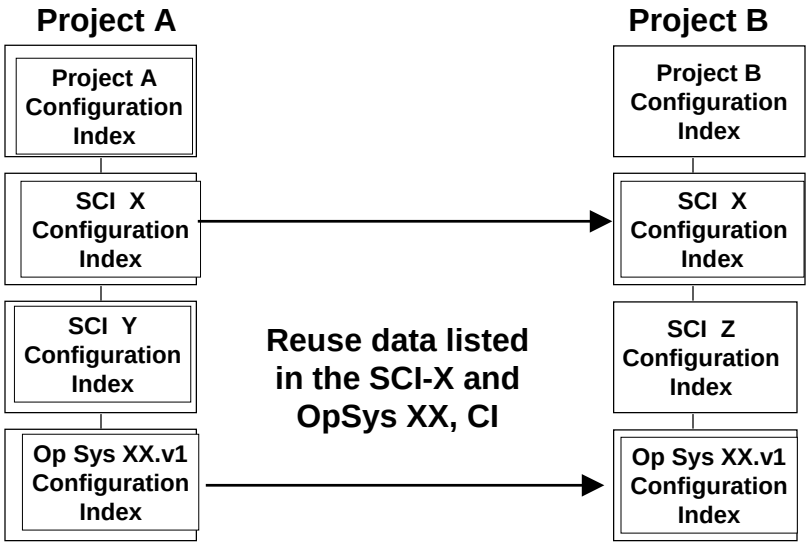
Order 8110.49 (Ch 12) Overview

- Based on Notice 8110.97
- Addresses reuse of software life cycle data within a company
- Outline:
  - 12-1: General
  - 12-2: Software Suitable for Reuse
  - 12-3: Safety Considerations
  - 12-4: Factors Affecting Reuse
  - 12-5: Reuse Approval Guidelines
- Good packaging is needed to effectively reuse software life cycle data.

Examples of Good Packaging

- Develop plans and standards to be as “generic” as possible, with project-specific information in the PSAC.
- Build and package the software so it can be used on multiple projects
- Tool qualification data separate for tools used on all software projects
- Make individual software configuration indices (SCIs) for components that may later be reused
- Design the software components for reuse (high cohesion, low coupling)

Conceptual Framework



Applicable Definitions

Original Certification Project	First use of the reusable software life cycle data in a completed cert project.
Subsequent Certification Project	Follow-on project that reuses software life cycle data from the original certification project.
Reuse	Subsequent use of unaffected, previously approved software life cycle data.
Certification Credit	Acceptance that a process, product, or demo meets the certification requirements.

Applicable Definitions (cont)

Software Life Cycle Data	Data produced during the software life cycle. Also known as the DO-178B, Section 11 data.
Configuration Item	1) One or more software components treated as a unit. 2) Software life cycle data treated as a unit.
Software Configuration Index	Identifies configuration of an item. Contains one or more configuration items.
Software Life Cycle Env. Index	Identifies configuration of the software life cycle environment.

Applicable Definitions (cont)

Software Plans & Standards	Data that directs the development & integral processes.
Software Tool	Computer program used to develop, test, analyze, produce, or modify another program or its documentation.
Tool Qualification	Process necessary to obtain cert credit for a tool.
Software Library	Collection of software and related data/ documents.

12-2: Software Suitable for Reuse

- **Software plans and standards**
- **Tool qualification data**
- **Software libraries**
- **Software requirements, design, code, verification procedures, and verification results.**
- **Configuration items**
- ***Basically: any unchanged software life cycle data***

12-3: Safety Considerations

- **FAA can approve for reuse if:**
  - **There is no adverse effect on original systems safety margins, <u>and</u>**
  - **There is no adverse effect on original operational capability UNLESS accompanied by justifiable increase in safety.**

12-3: Safety Considerations (cont)

- **FAA will not approve for reuse if reuse:**
  - **Adversely affects safety,**
  - **Exceeds a pre-approved range of data or parameters, <u>or</u>**
  - **Exceeds equipment performance characteristics.**

12-4: Factors Affecting Reuse

a. Any Section 11 data can be reused if:
- **It remains unchanged**
- **It is applicable to the project**
- **No safety issues exist**

b. In-service problems might limit reuse. Open problems reports should be analyzed prior to reuse

12-4: Factors Affecting Reuse (cont)

c. Assessment should be performed to show similarity of operational environment and safety assessment
  - Builds on a and b

12-5: Reuse Approval Guidelines

- Certification authority should ensure that:
  - Data to be reused is unchanged.
  - The software level is equivalent to (or less than) software level of the previous approval.
  - Range & data type of inputs are equivalent to previous approval.
  - Configuration items are used on the same target environment and in same operational way.

12-5: Reuse Approval Guidelines (cont)

- Certification authority should ensure that (cont):
  - Equivalent software/hardware integration and system testing conducted on same target and system as previous approval.
  - Applicant addressed safety considerations.
  - Reuse rationale is documented in "Additional Considerations" portion of the PSAC.

Summary of Chapter 12 of Order 8110.49

- Reuse of software life cycle data on multiple certification projects is feasible
- If a data item hasn't changed and is applicable for the current project, it is a candidate for re-use
- Present plan for reuse in PSAC and get early ACO agreement

Break

AC 20-RSC

Title: Reusable Software Components (RSC)
Target Release: Nov/Dec 2003
IVT Based on: Draft 9.2 (9/24/03)

SEE APPENDIX E

Goal: To be able to carry certification "credit" for reusable software component from one project to the next

Why Is This AC Needed?

- To support IMA concept, where reuse is critical
- To optimize use of FAA and applicant resources
- To provide guidance for third party manufacturers who may not have certification experience
- To ensure that all applicable DO-178B objectives are met for reusable components

Addressing Some Misconceptions About This AC

- It is not an “approval”
- It does not release the applicant from responsibility
- It will likely require additional resources from FAA and applicants on the first use of an RSC

Addressing Some Misconceptions About This AC (cont)

- It is not easy
- The certification authority may need to do additional review activity on an RSC if installation, safety, operational, performance, or functional issues exist
- An RSC acceptance letter does not mean all the DO-178B objectives of the RSC are met yet

AC Overview

- 1-4: Purpose, Background, Related Documents, & Document Overview
- 5: Discussion & General Guidelines
- 6: Guidelines for the RSC Developer
- 7: Data Supplied to RSC Integrator and/or Applicant
- 8: Guidelines for the Integrator and Applicant
- 9: Expectations from Certification Authorities on the First Use of the RSC

AC Overview (cont)

- **10: RSC Acceptance Letter**
- **11: Expectations from Certification Authorities on Subsequent Use of an Accepted RSC**
- **12: Common Software Reuse Issues**
- **13: Changes to RSCs**
- **14: Concurrent use of an RSC**
- **Appendix 1 – Definitions**
- **Appendix 2 – Acronyms**
- **Appendix 3 – Sample Format for RSC Table**

Important Definitions

- **<u>Reusable software component (RSC)</u> is the software code and its supporting DO-178B documentation being considered for reuse. It forms a portion of the software that will be implemented by the integrator/applicant.**

- **<u>Reusable software component developer</u> (RSCD) is the manufacturer of the reusable software component.**

Important Definitions (cont)

- Integrator is the manufacturer responsible for integrating the re-useable software component into the target computer and with other software components.
- Applicant is the manufacturer seeking certification or authorization of the overall system.

Software Reuse - **157**
October 29-30, 2003

1-4: Purpose, Background, Related Documents, &Document Overview

- Traditionally, software approval is at the system level.
- In the past no vehicle to carry certification credit across project boundaries existed.
- Purpose of this AC is to provide guidelines for allowing "credit for DO-178B objectives" across projects.

Software Reuse - **158**
October 29-30, 2003

1-4: Purpose, Background, Related Documents, &Document Overview (cont)

- **RSC Examples:**
  - **Operating Systems**
  - **Libraries**
  - **Loading Software**
- **AC ensures that all applicable DO-178B objectives are met for each use of the RSC.**
- **Guidelines are applicable within a company or across company boundaries.**

1-4: Purpose, Background, Related Documents, &Document Overview (cont)

- **Note: Concepts of component reuse document in this AC may apply to tools (see note in Section 2).**

Section 5: Discussion

STAKEHOLDERS

Integrator

Reusable Software Component Developer (RSCD)

Applicant

Certification Authorities

Note: Cert authorities may have more involvement than a traditional software development project for the initial component development.

Section 5: Discussion (cont)

a. **First acceptance of RSC is a "real" project.**
b. **Stakeholders must agree on reuse.**
c. **Each project is unique and might have different "credit".**
d. **Applicant is responsible for final certification.**
e. **Acceptance on one project doesn't guarantee acceptance on another. Installation, safety, operational, functional, and performance considerations must be considered on each project.**

Section 5: Discussion (cont)

f. International programs may require additional coordination.
g. Integrator/applicant required to address all objectives and coordinate communication.
h. Other guidance and regulations apply.
i. Discourages alternate means to DO-178B.

Overview of the Process

Stakeholders agree that reuse is a desirable & obtainable goal.

RSCD, integrator, & applicant plan for reuse.

RSCD, integrator, & applicant document reuse credit per objective.

PSAC reviewed & approved by cert authorities.

RSC developed per plans with cert authority oversight.

ACO writes acceptance letter for RSC to RSCD and applicant.

Same configuration & version of RSC used on other programs within limitations.

Section 6 – Guidelines for RSC Developer

- **Document Reuse intent in the PSAC**
  - **Reuse credit for each objective**
  - **Assumptions for each objective**
  - **Means of compliance for each objective**
  - **Remaining activities for the installer/applicant for each objective**
- **Appendix 3 contains an example format for documenting this information**

Example Approach (Appendix 3)

178B Obj #	*Obj Description*	*Credit Sought*	*Assumption*	*Means of Compliance for the Objective*	*Activities Remaining For Integrator/Applicant*
1-1	**Software development and integral processes activities are defined.**				
1-2	**Transition criteria, inter-relationships and sequencing among processes are defined.**				

Example Approach (cont)

- Document objective, credit sought, assumptions, and remaining activities in the PSAC and Accomplishment Summary.
- Address target dependencies.
- Address assumptions regarding requirements; particularly high-level requirements.
- Be specific and thorough.
- Obtain FAA input & agreement on proposals up-front.

Example Approach (cont) [Full Credit]

- Objective 1-1: Software development and integral processes activities are defined.
- Credit Sought: Full
- Assumptions: Plans are completed and unchanged for router.
- Remaining Activities: Applicant/integrator to complete LRU level plans, reference router plans/data, & consider reuse in "Additional Considerations"

Example Approach (cont) [Partial Credit]

- **Objective 2-1: High-level requirements are developed.**
- **Credit Sought: Partial**
- **Assumptions: Assuming high level requirements are document XXX, revision - and the LRU manufacturer uses those requirements.**
- **Remaining Activities: Because the high-level requirements actually exist at the LRU level, they cannot be fully implemented at the software component level. The applicant may reference and tie to the component-level high-level requirements as their own high-level requirements. If this occurred, the applicant would also need to verify the high-level functionality of these requirements in their system.**

See Table in Appendix F of Participant's Guide

Turn to Exercise 5 (Appendix C)

Section 6 – Guidelines for RSC Developer (cont)

- **Document safety-related issues**
- **Address common reuse issues (section 12)**
- **Coordinate plans with all stakeholders & follow them**
- **Submit SAS and SCI, with the completed compliance tables**
- **Submit data sheet to the FAA**
  - **See Appendix G for an example data sheet**

Section 6 – Guidelines for RSC Developer (cont)

- Supply data to the applicant to support the type design and continued airworthiness (per Section 7)
  - Turn to Section 7 of the AC to review the software life cycle data to be supplied to the applicant.

Section 8 - Guidelines for RSC Integrator/Applicant

- Integrate RSC data into the project data
- Specify the life cycle data needed from the applicant (per Section 7)
- Evaluate installation, safety, operational, performance, and functional issues of the RSC
- Coordinate & follow plans and standards
- Consider open problem reports and in-service problems of the RSC

Section 8 - Guidelines for RSC Integrator/Applicant (cont)

- Address common reuse issues listed in section 12 of AC and other issues
- Ensure assumptions made by the RSC developer are met in the application
- Complete the RSC objectives tables in the SAS
- Report in-service difficulties with the RSC
- Establish necessary legal agreements with the RSC developer to meet regulations

Section 9 – Expectations From Cert Authority on 1st Approval of the RSC

- Involvement with all stakeholders
- Involvement of technical experts, as needed
- Review plans of RSC developer and 1st applicant for consistency
- Perform on-site & desk reviews, as needed

Section 9 – Expectations From Cert Authority on 1st Approval of the RSC (cont)

- **Ensure that process is in place to support continued airworthiness**
- **Approve project, when objectives are satisfied**
- **Write letter for RSC developer explaining acceptance, limitations, etc. (per Section 10)**
  - **See Appendix H of participant's guide for an example**

Turn to Exercise 6 (Appendix C)

Section 11 – Expectations From Cert Authority on Subsequent Use of RSCs

- Review the acceptance letter
- Contact ACO engineer who did the original acceptance, if needed
- Ensure that the applicant follows the guidelines of this AC
- Ensure that installation, safety, operational, functional, and performance concerns are addressed in the subsequent reuse

Section 11 – Expectations From Cert Authority on Subsequent Use of RSCs (cont)

- Perform reviews of project plans and data
- Ensure consistency between RSC plans/data and applicant's plans/data
- Ensure that any RSC in-service problems do not affect safety of the subsequent reuse
- Inform original ACO of subsequent use/approval of RSC

Section 12 – Common Issues & Considerations

- Requirements definition
- Re-verification
- Interface
- Partitioning/Protection
- Data and Control Coupling
- Use of Qualified Tools
- Deactivated Code
- Traceability
- Robustness

Section 13 – Changes to RSCs

- When RSC is changed, cannot be reused without another reuse application.

Section 13 – Changes to RSCs (cont)

- **Change impact analysis should be performed on changes to RSCs**
  - **Info from Order 8110.49 on change impact analysis is repeated, because we could not reference the Order in the AC**

Section 14 – Concurrent Use of An RSC

- **RSC Developer Should Create a "Reuse Plan" including:**
  - **List of known applicants**
  - **Policy for addressing additional applicants in the future**
  - **Schedule for upcoming projects**
  - **Reuse approach**
  - **Software life cycle data being developed**
  - **Summary of unchanged data to be used by applicants and any user-specific data**

Section 14 – Concurrent Use of An RSC (cont)

- "Reuse Plan" includes (cont)
  - Suggestions for optimizing FAA and applicant resources. For example:
    - Suggested ACO to lead the effort (based on applicant locations and schedules)
    - Suggested approach for performing software reviews
  - List of affected applicants & ACOs
  - Plan for informing and keeping users up-to-date as the RSC develops

Section 14 – Concurrent Use of An RSC (cont)

- "Reuse Plan" should be coordinated with affected ACOs, applicants, and integrators.

Experiences Using the RSC Concept To Date

- Several Real-Time Operating Systems (RTOS)
- A Communication Stack
- A Piece of the NexComm System (the vocoder)
- A C++ Library

Software Reuse - **187**
October 29-30, 2003

Status of AC 20-RSC

- Have Addressed Public Comments
- Version 9.2 Went Forward For Final Coordination
- AC is in the Final Signature Process
- Should be Signed in the Nov/Dec timeframe

Software Reuse - **188**
October 29-30, 2003

Break

Other Reuse-Related Activities

- COTS Research
- OO Research and Handbook
- Software Service History Handbook
- Integrated Modular Avionics
- Tools Research and Reuse

COTS Research

COTS Research

- **As explained in Day #1, four phases of COTS research have been completed**
- **Reports may be found on the FAA's software web-site:**
  - **http://av-info.faa.gov/software**
- **The research is now focusing on the integration of RTOSs and other components**

OOT Research and Handbook

OOT Research Efforts

- **Industry-sponsored research in OOT started in 2000**
  - **Aerospace Vehicle Systems Institute (AVSI)**
  - **Created a Guide**
- **FAA-sponsored research in OOT started in 2000**
  - **Focusing on structural coverage issues**
  - **Through NASA Langley and Boeing**
  - **Report available on FAA's software web-site (http://av-info.faa.gov/software)**

OOT Research Efforts (cont)

- Both industry and FAA efforts revealed the need for specific guidance when using OOT in aviation products
- FAA is sponsoring a new task to consider the following OOT-related issues:
  - Data coupling and control coupling
  - Structural coverage at the object code level

OOTiA Workshops

- NASA and FAA sponsored two Object-Oriented Technology in Aviation (OOTiA) workshops
- Workshop #1 was held in April 2002
- Workshop #2 was held in March 2003
- Both workshops were intended to get government, industry, and academia together to consider OOTiA

OOTiA Workshops (cont)

- **The result of the OOTiA workshops and the OOTiA team efforts will a 4-volume handbook. Due June 2004.**

Volume 4: *Certification Practices*

Volume 3: *Best Practices*

Volume 2: *Considerations and Issues*

Volume 1: *Handbook Overview*

OOTiA Workshops (cont)

- **Volume 1: Handbook Overview**
  - **Target Audience: All Handbook users**
  - **Provides background and foundational information needed to use all other volumes**

OOTiA Workshops (cont)

- Volume 2: Considerations and Issues
  - Target Audience: Project planners, decision makers, certification authorities
  - Poses questions to be answered before committing to OOT
  - Presents concerns raised about OOT that are relevant to certification and safety without discussing approaches for addressing these concerns
  - Categorizes, summarizes, and discusses key issues
  - Provides issue rationale and ties to DO-178B life cycle processes

OOTiA Workshops (cont)

- Volume 3: Best Practices
  - Target Audience: Developers, certification authorities
  - Identifies best practices to safely implement OOT in aviation by providing some known ways to address the issues documented in Volume 2

OOTiA Workshops (cont)

- Volume 4: Certification Practices
  - Target Audience: Certification authorities, designees
  - Provides an approach for certification authorities and designees to ensure that OOT issues are addressed

OOTiA Workshops (cont)

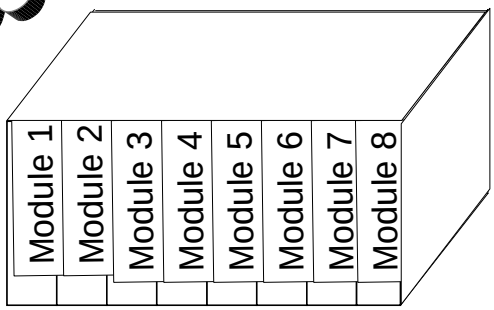
- For more info on OOTiA:
  - Go to: http://shemesh.larc.nasa.gov/foot
  - Contact Barbara Lingberg at: Barbara.Lingberg@faa.gov

Software Service History

SSH Research

- **As explained on Day #1, a SSH Handbook has been created and is on the FAA's software web-site.**
- **Follow-on research has been carried out on the WAAS ground-based program (report to come soon).**
- **FAA hopes to sponsor additional research in the area of service history for <u>hardware</u> tools.**

Integrated Modular Avionics (IMA)



What is IMA?

- **Defining "IMA" is difficult**
- **The RTCA modular avionics team created the following definition:**
  - **Modular avionics is defined as a shared set of flexible, reusable, and interoperable hardware and software resources that create a platform that provides services, designed and verified to a defined set of safety and performance requirements, to host applications performing aircraft-related functions.**

IMA-Related Efforts

- **RTCA Special Committee (SC) 182**
  - **Created DO-255 (Avionics Computer Resource Requirements)**
- **FAA's IMA Team**
  - **Developed "TSO-C153, INTEGRATED MODULAR AVIONICS HARDWARE ELEMENTS" (May 2002)**
  - **Developed AC 20-145: *"GUIDANCE FOR INTEGRATED MODULAR AVIONICS (IMA) THAT IMPLEMENT TSO-C153 AUTHORIZED HARDWARE ELEMENTS"* (February 2003)**

IMA-Related Efforts (cont)

- **RTCA Special Committee (SC) 200**
  - **Developing IMA Guidance material**
  - **Joint with EUROCAE Working Group #60**
- **The "Hardware Element" approach seeks to reuse hardware elements**
- **The SC-200/WG-60 effort is striving to reuse "module qualification" packages**

For More Info IMA TSO and AC

- Read TSO-C153 and AC 20-145 at
  http://av-info.faa.gov/software
- Review the IMA IVT from October 2002

Software Reuse - **209**
October 29-30, 2003

RTCA Special Committee #200 (SC-200) & EUROCAE Working Group #60 (WG-60)

Software Reuse - **210**
October 29-30, 2003

SC-200/WG-60 Schedule

- 3/02 - SC-200 Approved by RTCA
- 5/02 - First SC-200 Meeting
- 8/02 - Became joint with EUROCAE Working Group #60 (WG-60)
- 3/04 – Goal for Draft Guidance Document
- 10/04 – Goal for Final Guidance Document

SC-200/WG-60 Highlights

- Builds on FAA's TSO and AC
- Builds on DO-255 (Avionics Computer Resource)
- Uses the "Module Qualification" concept

Document Overview

- Section 1: INTRODUCTION
  - Purpose
  - Scope
  - Background
  - Stakeholders
  - Relationship to other documents
  - References
  - How to use the document

Document Overview (cont)

- Section 2: MODULAR AVIONICS (MA)
  - System description & architecture
  - Key characteristics

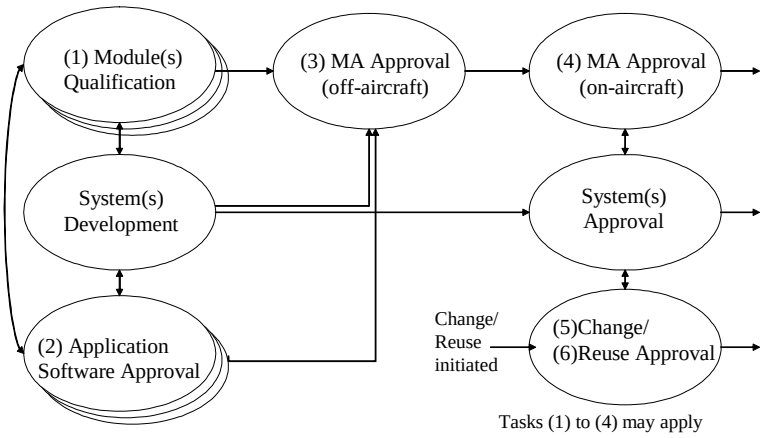
Document Overview (cont)

- **Section 3: MA-Specific Design and Integration CONSIDERATIONS**
  - **Design objectives**
  - **Partitioning & resource management**
  - **Health monitoring & fault management**
  - **Configuration**
  - **Integration**
  - **Shared databases for applications**

Document Overview (cont)

- **Section 4: MODULAR AVIONICS CERTIFICATION CONSIDERATIONS**
  - **System development**
  - **Six tasks (see next slide)**
  - **Safety assessment**
  - **Design assurance (software, hardware, environment, security)**
  - **Verification & Validation**
  - **Integration**
  - **Configuration management**
  - **Quality assurance**
  - **Certification Liaison**

Six Modular Avionics (MA) Tasks



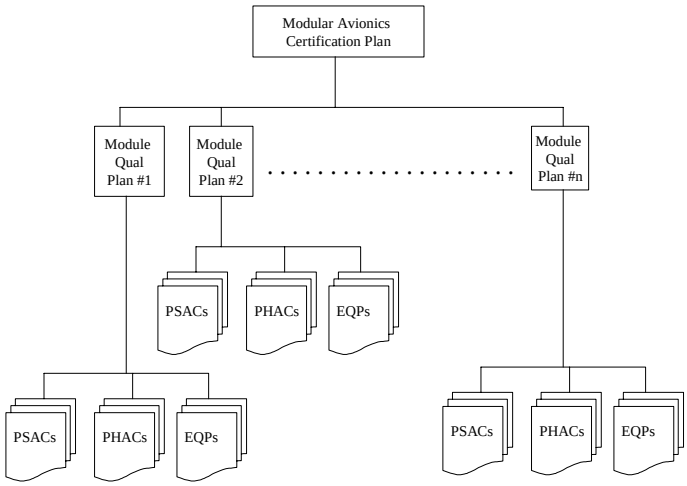
Document Overview (cont)

- **Section 5: SIGNIFICANT ISSUES**
  - **Flight operations**
  - **Installation**
  - **Continued airworthiness**
  - **Human factors**

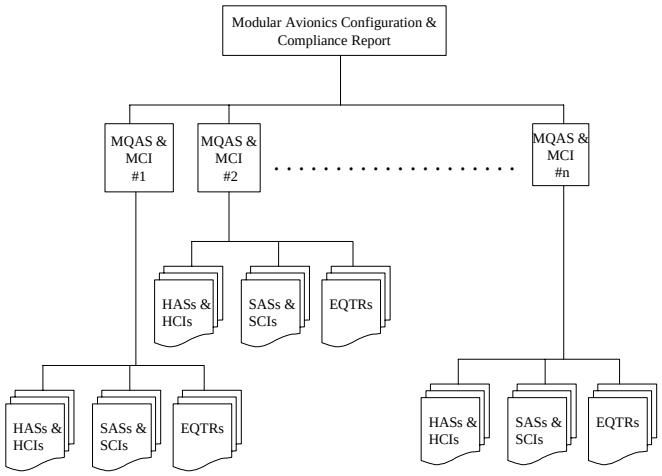
SC-200/WG-60 “Module Qualification” Concept

- “Module” is software, hardware, or a combination of software and hardware.

SC-200/WG-60 “Module Qualification” Concept (cont)



SC-200/WG-60 “Module Qualification” Concept (cont)



Software Tool Research and Reuse

Software Tool Research
Development Tools

- **Embry-Riddle through FAA's Airworthiness Assurance Center of Excellence**
- **Objectives:**
  - **To establish a base for assessment of software development tools**
  - **To create a taxonomy and a set of criteria/guidelines for tool selection and qualification**
  - **To perform an experiment collecting data using selected development tools**
- **One more year remaining.**

Software Tool Research
Verification Tools

- **NASA Langley and Boeing Wichita**
- **Purpose is to develop effective criteria for evaluating structural coverage analysis tools for use on a DO-178B project.**
- **One more year remaining.**

Software Tool Reuse

- Order 8110.49, Chapter 12 includes reuse of tool qualification data within a company
- AC 20-RSC has a note which allows the AC concept to apply to tools on a case-by-case bases
- CAST is working on a tool reuse paper, which will likely be the starting point for an update to AC 20-RSC or a new AC on tool reuse

Keys To Reuse Acceptance

Keys To Reuse Acceptance

- Ensure that communication among all stakeholders is established.
- Ensure that the users (aircraft, engine, and avionics manufacturers) have the necessary data and expertise to properly use the software.
- Ensure that all DO-178B objectives will be met in the certified or authorized project.

Keys To Reuse Acceptance (cont)

- Evaluate installation, safety, operational, functional, and performance concerns and responses on all uses of reused software.
- Ensure that the developer has truly planned for reuse rather than salvaging.
- Use additional resources to ensure that the first acceptance is done well.

Keys To Reuse Acceptance (cont)

- **Ask for help from specialists, when needed.**
- **Ensure that the common reuse concerns documented in section 12 of AC 20-RSC are addressed, as well as any project-specific concerns.**

Summary

- Order 8110.49 addresses reuse of software life cycle data within a company
- AC 20-RSC addresses reuse of software components across company boundaries (when the components are unchanged)
- COTS Research has been carried out and focuses on the real-time operating system.
- OOT research has provided input for the handbook
  - OOTiA handbook slated for completion in June 2004

Summary (cont)

- Service History Research & Handbook are available to support reuse but is a difficult case to make.
- IMA intends to reuse data
  - Hardware element TSO
  - "Module qualification" concept being proposed by SC-200/WG-60
- Reuse policy and guidance may apply to tool qualification as well

Summary (cont)

- **Software can be safely reused if it is well planned and carefully implemented.**

LIST OF ACRONYMS USED IN THE IVT

Appendix B

Acronyms Related "Software Reuse in Airborne Systems" IVT

AC	Advisory Circular
ACO	Aircraft Certification Office
AD	Airworthiness Directive
API	Application Programmer Interface
ARP	Aerospace Recommended Practice
ASTC	Amended Supplemental Type Certificate
ATC	Amended Type Certificate
CAST	Certification Authorities Software Team
CFR	Code of Federal Regulations
CMR	Certification Maintenance Requirement
COTS	Commercial-off-the-shelf
CSTA	Chief Scientific and Technical Advisor
DER	Designated Engineering Representative
EQP	Environmental Qualification Plan
EQTR	Environmental Qualification Test Report
FAA	Federal Aviation Administration
I/O	Input/Output
IMA	Integrated Modular Avionics
IVT	Interactive Video Teletraining
HAS	Hardware Accomplishment Summary
HCI	Hardware Configuration Index
LRU	Line Replaceable Unit
MA	Modular Avionics
MCI	Module Configuration Index
MQAS	Module Qualification Accomplishment Summary
NASA	National Aeronautics and Space Administration
OO	Object-Oriented
OOA	Object-Oriented Analysis
OOD	Object-Oriented Design
OOP	Object-Oriented Programming
OOT	Object-Oriented Technology
OOTiA	Object-Oriented Technology in Aviation
OOV/T	Object-Oriented Verification/Testing
PR	Problem Report
PHAC	Plan for Hardware Aspects of Certification
PSAC	Plan For Software Aspects Of Certification
REBOOT	Reuse Based on Object-Oriented Technology
RMM	Reuse Maturity Model
RSC	Reusable Software Component
RTOS	Real-Time Operating System
SAS	Software Accomplishment Summary
SC	Special Committee
SCI	Software Configuration Index
SCM	Software Configuration Management

SCMP	Software Configuration Management Plan
SDP	Software Development Plan
SQA	Software Quality Assurance
SQAP	Software Quality Assurance Plan
SSA	System Safety Assessment
SSH	Software Service History
STC	Supplemental Type Certificate
SVP	Software Verification Plan
SW	Software
TC	Type Certificate
TSO	Technical Standard Order
WAAS	Wide Area Augmentation System
WG	Working Group

EXERCISES TO BE USED IN THE IVT

Appendix C

Exercise 1

Describe some situations in your job where you have seen salvaging and reusing. List the situations below and be prepared to discuss with the class.

Salvaging	Reusing

Exercise 2

<u>Scenario</u>: Assume that you are an ACO engineer involved in a project that will use a COTS RTOS.

<u>Question</u>: What are some of the things you would do in this situation?

Exercise 3

Scenario: Assume that you are an ACO engineer involved in a project that proposes to use product service history.

Question: What are some of the things you would do in this situation?

Exercise 4

Question: Given the list of "common certification concerns with reuse" what do you think are some ways to address these concerns in a safe manner?

Exercise 5

Scenario: Assume that you are an ACO engineer working with a RSC developer of an operating system that is designed to be reusable.

Exercise: Using the tables in Appendix F of your Participant's Guide, list some of the things you would consider as a "regulator".

Exercise 5 (cont)

Considerations (list here):

Exercise 6

<u>Question</u>: Review the sample acceptance letter in Appendix H and compare it with the suggested items listed in Section 10 of AC 20-RSC. Are there any additional things that you would include in the letter? If so, list them below:

Chapter 12 of Order 8110.49

APPROVING REUSED SOFTWARE LIFE CYCLE DATA

Appendix D

FROM ORDER 8110.49, SOFTWARE APPROVAL GUIDELINES

CHAPTER 12. APPROVING REUSED SOFTWARE LIFE CYCLE DATA

12-1. GENERAL. This chapter provides guidelines for determining if software life cycle data, produced and approved for one certification project, can be approved on a follow-on certification project. Approval for reuse could minimize the amount of rework while maintaining an equivalent level of design assurance.

12-2. SOFTWARE SUITABLE FOR REUSE.

a. If properly planned and packaged, software life cycle data can be reused from one project to the next, with minimal rework. For example, the software plans, requirements, design, and other software life cycle data (as documented in a Software Configuration Index) for a Global Positioning System (GPS) may originally be approved on GPS #1 (the original certification project) and reused on GPS #2 (the subsequent certification project). Sample items suitable for reuse include:

(1) Software plans and standards. These include software undergoing non-substantive changes, such as:

- Program name,
- Name change due to consolidations or mergers, and
- Configuration changes for reasons other than design changes (for example, document format change, drawing modifications, or documentation system changes).

(2) Tool qualification data. The FAA can approve reuse, if the tool is used exactly as specified in the qualification approval as part of the original certification, and the applicant has access to the tool qualification data. This is true even if some of the features were qualified but not used during the original certification. The applicant should ensure that the same version of the tools is being used as that supported by the qualification data. The FAA will not approve reuse if the applicant uses additional or different tool functionality than was previously qualified.

(3) Software libraries. The FAA can approve library sets in the original certification project if the library set is used identically (that is, same library functions are used the same way).

(4) Software requirements, design, code, verification procedures, and verification results. The FAA may approve these for reuse after the applicant makes a thorough change impact analysis. This is to confirm that the requirements, design, code,

procedures, and so forth are unaffected and unchanged from the previous certification effort.

(5) Configuration items. These may be approved for reuse in their entirety, if the certification authority and DERs use paragraphs 12-3 through 12-5 of this chapter to make the determination, and the configuration of the software life cycle data has not changed. Configuration item requirements verified at a higher level (that is, system level) should be identified in the original configuration and reverified before reuse.

b. Projects not using RTCA/DO-178B may have additional considerations not documented in this chapter. Certification authorities should evaluate them on a case-by-case basis. The applicant should contact their local certification authority for guidance. The certification authority should coordinate with the CSTA for Aircraft Computer Software, the appropriate Directorate, and/or AIR-120, as necessary.

12-3. SAFETY CONSIDERATIONS. If the FAA finds software life cycle data acceptable for reuse, no further design approval is required. Figure 12-1 illustrates the considerations that govern whether the FAA will approve software reuse.

Figure 12-1. Reuse Approval Considerations

FAA may approve for reuse if:	1. There is no adverse effect on original system safety margins, **and** 2. There is no adverse effect on original operational capability UNLESS accompanied by a justifiable increase in safety.
FAA will NOT approve for reuse if the reuse:	1. Adversely affects safety, 2. Exceeds a pre-approved range of data or parameters, **or** 3. Exceeds an equipment performance characteristic.

12-4. FACTORS AFFECTING REUSE.

a. Any of the software life cycle data in Section 11, RTCA/DO-178B is suitable for reuse. To meet the guidelines in paragraph 12-5 of this chapter, the software life cycle data should be unchanged, and should apply to the project for which reuse is being considered.

b. In-service problems with previous applications can limit reuse. There may be Airworthiness Directives or a manufacturer's unresolved problem reports with the previously approved system. The applicant needs to analyze all open manufacturer's problem reports to ensure that the reusable portion of the new software is not affected. If the reusable portion of the new software is affected, changes to correct that software life cycle data should be made or the software should not be used.

c. Applicants should determine if the software data apply to the subsequent project's development by assessing the similarity of both the operational environment and the software development environment. They should:

(1) Assess the operational environment by evaluating the end-to-end performance requirements and the operational safety assessment.

(2) Refer to the Software Life Cycle Environment Configuration Index in Section 11.15, RCTA/DO-178B, when assessing the software development environment.

(3) Demonstrate that operational and development environments are the same, or demonstrated to produce identical results as the previous certification.

(4) Assess any outstanding problem reports.

12-5. REUSE APPROVAL GUIDELINES.

a. The certification authority should ensure that the applicant has met the following guidelines before granting certification credit for reused software life cycle data:

(1) The software life cycle data have not changed since its previous approval.

(2) The software level of the software application(s) is equal to (or less than) the software level of the original certification effort.

(3) The range and data type of inputs to the configuration item are equivalent to its approved predecessor.

(4) The configuration item is embedded on the same target computer and is used the same way operationally as the original certification project.

(5) Equivalent software/hardware integration testing and system testing were conducted on the same target computer and system as in the original certification project.

(6) The applicant followed the safety considerations and reuse factors in paragraphs 12-3 and 12-4 of this chapter.

(7) The software life cycle data and the rationale for reuse of each item are documented in the "Additional Considerations" portion of the PSAC. The applicant's PSAC should include method of use, integration, and documentation for the reused configuration item. The PSAC should be submitted as early as possible in the development program. The applicant should also document all references to the project previously certified and the project number, as applicable, in the PSAC.

b. The certification authority responsible for the subsequent certification should review the PSAC and notify the applicant whether the proposal is acceptable or not (with appropriate rationale).

ADVISORY CIRCULAR 20-RSC

REUSABLE SOFTWARE COMPONENTS

Draft 9.2

Appendix E

U.S. Department
of Transportation
Federal Aviation Administration

Advisory Circular

AC 20-RSC

REUSABLE SOFTWARE COMPONENTS

DRAFT 9.2 – 9/24/03

Date: XXXXXXXX

TABLE OF CONTENTS

SECTION **PAGE**

1. PURPOSE. 1

2. BACKGROUND. 2

3. RELATED DOCUMENTS. 2

4. DOCUMENT OVERVIEW. 3

5. DISCUSSION. 4

6. GUIDELINES FOR THE RSC DEVELOPER. 5

7. DATA SUPPLIED TO THE RSC INTEGRATOR AND/OR APPLICANT. 8

8. GUIDELINES FOR THE INTEGRATOR AND APPLICANT USING THE RSC. 10

9. EXPECTATIONS FROM CERTIFICATION AUTHORITIES ON THE FIRST USE OF AN RSC. 12

10. THE RSC ACCEPTANCE LETTER. 13

11. EXPECTATIONS FROM CERTIFICATION AUTHORITIES ON THE SUBSEQUENT USE OF AN ACCEPTED RSC. 14

12. COMMON SOFTWARE REUSE ISSUES. 15

13. CHANGES TO REUSABLE SOFTWARE COMPONENTS. 19

14. CONCURRENT USE OF AN RSC. 22

APPENDIX 1 – DEFINITIONS OF TERMS A1-1

APPENDIX 2 - ACRONYMS A2-1

APPENDIX 3 – SAMPLE FORMAT FOR RSC DEVELOPER'S TABLE A3-1

DRAFT

U.S. Department of Transportation
Federal Aviation Administration

Advisory Circular

Subject: **REUSABLE SOFTWARE COMPONENTS**	**Date:** XXXXXXXX	**AC No:** 20-RSC
	Initiated By: AIR-120	**Change:**

1. PURPOSE.

a. This advisory circular (AC) provides one acceptable means of compliance, but not the only means, for use by reusable software component (RSC) developers, integrators, and applicants to gain Federal Aviation Administration's (FAA) "acceptance" of a software component that may be only a part of an airborne system's software applications and intended functions. Like all advisory material, this AC is not mandatory and does not constitute a regulation. Because the means of compliance presented in this AC is not mandatory, the term "must" used herein applies only to the applicants, integrators, and RSC developers who choose to follow the method prescribed in this AC.

b. This AC also shows a means to get credit for the reuse of a software component in follow-on systems and certification projects, including receiving "full credit" or "partial credit" for compliance to the objectives of RTCA/DO-178B, *Software Considerations in Airborne Systems and Equipment Certification*. When all stakeholders comply with this AC and no installation, safety, operational, functional, or performance concerns are identified by the FAA (or authorized designee), the FAA may grant acceptance for the RSC. This acceptance is accomplished by the issuance of an FAA RSC acceptance letter; the letter will not be written until a certification or authorization is granted for a product or equipment using the RSC. If the RSC is unchanged and meets the limitations stated in the RSC acceptance letter, it may be reused without additional FAA review of the RSC data, assuming no safety, installation, operational, functional, or performance concerns are identified in the subsequent application(s). This AC requires that the RSC being considered for acceptance have its own set of software life cycle data.

c. This AC applies to the "acceptance" of an RSC to support the approval of airborne systems and equipment related to the overall software aspects of those systems (including the RSC integrated with the other system's software and demonstrated to fully comply with the applicable regulations, guidance, and RTCA/DO-178B objectives) related to type

DRAFT

certificates (TC), supplemental type certificates (STC), amended supplemental type certificates (ASTC), amended type certificates (ATC), and Technical Standard Order (TSO) authorizations. For TSO authorized articles that are highly integrated into the aircraft or that have aircraft dependencies, the RSC acceptance letter will typically not be granted until the TSO authorized article and the RSC have received installation approval as part of a TC, STC, ASTC, or ATC. This is necessary because of the aircraft system's complexity and aircraft-specific dependencies on installed systems and equipment.

2. BACKGROUND.

a. Economic incentives and advances in software component technology have made it desirable to develop an RSC that can be integrated into a number of systems' target computers and environments with other system software applications, as determined by the integrator and/or applicant. In these cases, a developer of an RSC may partially satisfy the applicable RTCA/DO-178B objectives, while the integrator and/or applicant are responsible for completing and demonstrating the compliance for the integrated software package, systems aspects, and aircraft certification compliance activities. Examples of potential RSCs include software libraries, operating systems, and communication protocols.

b. The guidance in this AC is needed to ensure that all applicable RTCA/DO-178B objectives are met for systems that use RSCs.

NOTE: The reuse concept described in this AC may be applicable to verification and development tools; however, the details of each reusable tool qualification project must be discussed with the FAA. Tools differ from airborne software; therefore, there are some additional concerns to be addressed, when attempting to reuse tool qualification data. The FAA plans to specifically address tool reuse in future guidance.

3. RELATED DOCUMENTS.

a. Code of Federal Regulations (CFR). Title 14 of the Code of Federal Regulations parts 21, 23, 25, 27, 29, 33, and 35 are referenced throughout this AC. Copies of the CFRs are available from the FAA website at http://www.airweb.faa.gov/rgl.

b. FAA Advisory Circulars. AC 20-115B, *RTCA, Inc., Document RTCA/DO-178B*, dated January 11, 1993, offers a way to obtain FAA approval of software. The intent of this RSC AC is to provide guidance for applicants that use RTCA/DO-178B as their means of showing compliance to the regulations for software components. This AC supplements RTCA/DO-178B and AC 20-115B, for accepting compliance demonstration for some of the RTCA/DO-178B objectives for individual components of a system's software application and functions. If an applicant proposes a means of compliance other than RTCA/DO-178B, the FAA will decide whether this AC applies and whether additional policy or guidance is warranted. You can obtain copies of this AC, AC 20-115B, and other ACs from the FAA website at http://www.airweb.faa.gov/rgl.

DRAFT

c. FAA Policy Documents. Federal Aviation Administration (FAA) Order 8110.4, *Type Certification* (as amended), and Order 8110.49, *Software Approval Guidelines*, are relevant to this AC. You can obtain copies of orders from the FAA website at http://www.airweb.faa.gov/rgl.

d. RTCA, Inc. Documents. You may purchase copies of RTCA documents from RTCA, Inc., 1828 L Street, NW, Suite 805, Washington, D.C. 20036. Alternatively, copies may be purchased on-line at http://www.rtca.org/. RTCA documents referenced in this AC are:

(1) RTCA, Inc., Document RTCA/DO-178B, *Software Considerations in Airborne Systems and Equipment Certification*, dated December 1, 1992.

(2) RTCA, Inc., Document RTCA/DO-248B, *Final Report for Clarification of DO-178B Software Considerations in Airborne Systems and Equipment Certification*, dated October 12, 2001.

e. SAE Documents. You may purchase copies of Society of Automotive Engineers (SAE) documents from SAE, 400 Commonwealth Drive, Warrendale, PA 15096-0001. Or, copies may be purchased on-line at http://www.sae.org/. The following SAE documents are relevant to this AC:

(1) Aerospace Recommended Practice (ARP) 4754, *Certification Considerations for Highly-Integrated or Complex Aircraft Systems*.

(2) ARP4761, *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*.

4. DOCUMENT OVERVIEW. The following sections are included in this AC:

a. Sections 1 through 4 establish the context for this AC by providing background and introductory information.

b. Section 5 provides information and general guidelines for RSC acceptance.

c. Sections 6 through 8 provide guidelines for the RSC developers, integrators, and applicants, when developing or using an RSC.

d. Sections 9 through 11 provide typical activities that the RSC developers, integrators, and applicants can expect from the certification authorities for the first acceptance of an RSC and its subsequent use.

e. Section 12 discusses common issues that must be addressed when developing and using RSCs. These issues may affect multiple RTCA/DO-178B objectives. Section 12 does

not present an exhaustive list of issues that may arise, since each project will have its own specific issues.

f. Section 13 addresses changes to an RSC.

g. Section 14 considers concurrent uses of an RSC.

h. Appendix 1 defines the terms used in this AC. **Please review this appendix prior to reading the AC in order establish a consistent terminology.**

i. Appendix 2 lists the acronyms used in this AC.

j. Appendix 3 provides a sample format of an RSC developer's table.

5. DISCUSSION AND GENERAL GUIDELINES.

a. The first acceptance of an RSC must be performed during an actual project (such as, a TC, ATC, STC, ASTC, or TSO authorization project). This may require extra resources from the RSC developer, the integrator, the applicant, and the certification authority. Subsequent acceptance of the RSC for a different system or project will likely require less effort and resources, if the guidelines in this AC are followed.

b. This reuse guidance applies only when all the stakeholders (the applicant, integrator, RSC developer, and certification authority) agree that the software component is reusable. The RSC Plan for Software Aspects of Certification (PSAC) and the system-level PSAC of the first applicant are the recommended vehicles for documenting the agreement of the proposed means of compliance for the RSC to this AC in the context of the system and for defining the communication channels and roles among stakeholders. Agreeing on the reuse concept is important because the first applicant will likely use additional resources to qualify the component as reusable. If there is no agreement, then the traditional approach to software development and approval must be followed for all software in the system (see Section 6 of this AC).

c. Each RSC developer's project will have different limitations, needs, and issues. For example, one developer may package the software life cycle data so it fully satisfies a particular objective of RTCA/DO-178B. Another RSC developer may only partially satisfy that same objective. This may be due to some project-specific issues, or additional coordination with the integrator to augment the efforts of the RSC developer. Sections 6 through 8 of this AC guide the RSC developer, integrator, and applicant. The guidelines are meant to be flexible enough to satisfy the multiple needs of the RSC developer, integrator, and applicant. However, the guidelines are also detailed enough to ensure that relevant certification, compliance, and safety issues are addressed.

d. Applicants are responsible for submitting compliance data, coordinating, and communicating with certification authorities, and performing the certification liaison process for the project. However, there may be some communication between the

certification authority and the RSC developer (with the applicant's involvement) for the reuse aspects of the project.

e. It should be noted that acceptance of an RSC for one project does not guarantee acceptance on a subsequent project. Installation, safety, operational, functional, and performance considerations must be considered on each project. If concerns arise in any of these areas, certification authorities may need to reassess RSC life cycle data. Additionally, the compliance to all applicable RTCA/DO-178B objectives, guidance, and regulations must be addressed by every applicant on their particular project.

f. This AC was coordinated with international certification authorities; however, international programs or approvals may require additional activities not addressed in this AC. International projects should be closely coordinated with all applicable certification authorities.

g. The integrator and applicant should be aware that it is unlikely that an RSC can satisfy all of the objectives of RTCA/DO-178B and are advised that they may need validate RSC developer claims and provide additional resources for demonstrating compliance of systems containing RSCs. The integrator and applicant should also be aware that the communication paths and division of responsibilities can be complex, when using an RSC.

h. The integrator and applicant should also be aware that there are other regulations, guidance, and agreements that may be applicable for their system and its aircraft installation approval beyond the guidance of RTCA/DO-178B. These may be dependent on the date of application for the certification, the type of system which they are proposing, the introduction of novel design or technology or methods, or other factors. The applicant is responsible for demonstrating compliance of all components of their system, including RSCs.

i. It is recommended that the RSC developer, integrator, and applicant not propose alternative means of compliance to the objectives of RTCA/DO-178B for the software aspects of the initial approval or subsequent certification approvals of systems containing RSCs. As described in Section 3.b of this AC, if an alternate means is proposed, this AC may not be applicable.

6. <u>GUIDELINES FOR THE RSC DEVELOPER</u>. Prior to issuance of this AC, there were no procedures for RSC developers to directly transfer their accepted data from one project to the next and across company boundaries. Traditionally, RSC developers provided substantiation in one of two ways. First, by resubmitting the RSC data package and repeating the work for each system's application. Second, by providing traceability through the TC, ATC, STC, ASTC, or TSO approval back to the desired data and defending the validity of their processes and data from the original approval basis to the new approval basis for each system. This AC addresses the reuse of software components and software life cycle data across company boundaries. The RSC developer must do the following:

DRAFT

a. Produce a PSAC for the RSC as early as possible in the project. The RSC PSAC must:

- Include the information discussed in Section 11.1 of RTCA/DO-178B.
- Detail the RSC developer's plans for satisfying each applicable RTCA/DO-178B objective.
- Identify which objective(s) will not be satisfied and which objective(s) will be partially satisfied by the RSC developer.
- Explicitly state the RSC developer's agreement that the RSC is being developed with the intent to reuse it in future projects.
- State the intent to comply with this AC.
- Define the failure conditions, safety features, protection mechanisms, architecture, limitations, software level(s), interface specifications, and intended usage of the RSC.
- Provide a description of the proposed certification liaison process (including communication and coordination focal points) to all involved stakeholders.

b. Consider and address, as applicable, the common reuse issues documented in Section 12 of this AC.

c. For each RTCA/DO-178B objective applicable to the software level, document the information listed in items (1) to (4) below (in the RSC PSAC) with sufficient detail for certification authority concurrence and integrator and/or applicant usage of the RSC. The RSC developer may include this information in a table with columns for the objective reference, objective description, amount of credit being sought (full, partial, or no credit), assumptions, means of compliance, and remaining activities to be completed by the integrator and/or applicant (see a sample format in Appendix 3). Since resource-specific, target computer-specific, and system-specific issues may be uncertain early in the project, the RSC PSAC may list preliminary information that will be updated in RSC PSAC revisions and the RSC Software Accomplishment Summary (SAS), when the RSC is completed. Some reuse details may not be finalized until the end of the project. The following information must be thoroughly documented for each applicable RTCA/DO-178B objective for review by certification authorities (and authorized designees) and for usage by integrators and/or applicants:

(1) Credit being sought for the objective. The RSC PSAC or referenced document must specify if full, partial, or no credit is being sought for the objective. Full credit is defined as being able to completely satisfy an objective using the RSC data package and demonstrations that all associated assumptions are valid. If additional activity is required by the integrator, then full credit cannot be claimed. This is true even when the activities are fully specified by the RSC developer. Additionally, if the assumptions are not satisfied by the integrator or applicant, no credit can be obtained.

(2) Assumptions of the RSC developer on the behavior of the RSC users. Provide sufficient justification to ensure that the original acceptance is valid if the assumptions are satisfied. For example, the RSC developer may assume that the source

code, compiler type, and compiler options will remain the same. If, however, an integrator or applicant does not meet these assumptions, reuse of the applicable objective credit will not be allowed.

(3) Means of compliance for the objective. The RSC PSAC and SAS must document what software life cycle data supports compliance for each applicable objective (document titles, version numbers, and/or a description of the type of data to be provided as evidence of compliance).

(4) Activities remaining for the integrator and/or applicant. The RSC PSAC and SAS must document what an applicant and/or integrator must do to fully satisfy any partial or unsatisfied objectives.

d. Document the following safety-related items in the RSC PSAC and RSC SAS:

(1) The software level(s) for the RSC,

(2) An analysis of all interfaces and configurable parameters, which describes the functional and performance effects of these parameters on the user and any mitigations required by the user to ensure proper operation,

(3) Architectural and design features supporting any portion of the safety analysis, partitioning, or other protection strategies,

(4) Any safety, operational, functional, or performance assumptions that support the use of the RSC (see Section 6.e below), and

(5) Any new or novel concepts, methods, and technologies to be used in developing the RSC.

e. Additionally, the RSC developer must also produce an analysis of the RSC's behavior that could adversely affect the users' implementation (for example, a vulnerability assessment, partitioning analysis, hardware failure effects, requirements for redundancy, data latency, and design constraints for correct RSC operation). The analysis may be used to support the integrator and/or applicant's safety analysis.

f. Obtain agreement (as early as possible) by all stakeholders for the first application by coordinating the RSC PSAC, any other RSC plans (e.g., Software Development Plan (SDP), Software Verification Plan (SVP), Software Quality Assurance Plan (SQAP), and Software Configuration Management Plan (SCMP)), and software development standards (that is, requirements, design, and coding standards) with the certification authority, designees (if delegated), and the applicant and/or integrator.

g. Develop the RSC in compliance with the approved plans. As previously stated, the RSC developer must produce the RTCA/DO-178B software life cycle data and documentation identified in Section 7 of this AC for the RSC (such as, plans, standards,

DRAFT

development data, verification data, quality assurance records, and configuration management records).

h. Inform the certification authority, designees (if delegated), integrator, and applicant of both development progress and any deviations from plans, to allow for timely reviews and adjustments as necessary.

i. Submit the RSC Software Configuration Index (SCI) and the RSC SAS to the certification authority through the project applicant, when completed. The RSC SAS must include or refer to the software life cycle data of RTCA/DO-178B, Section 11, and the information discussed in Section 6 of this AC.

7. DATA SUPPLIED TO THE RSC INTEGRATOR AND/OR APPLICANT. The RSC developer must supply the appropriate software life cycle data to the integrator and/or applicant to support acceptance of the RSC in the context of the software aspects of certification of the airborne system(s) using the RSC. Typically, the RSC developer supplies the following data to the RSC integrator and/or applicant and to the certification authority on request (all except for item i below):

a. The type design data listed in Section 9.4 of RTCA/DO-178B for the RSC (that is, Software Requirements Data, Design Description, Source Code, Executable Object Code, SCI, and SAS).

b. The RSC PSAC, which identifies the credit sought for each RTCA/DO-178B objective.

c. Interface description data (for example, interface control document and porting guide). The interface description data includes any hardware and software resource requirements (such as, timing and memory) and applicable analyses, verification procedures, and verification cases.

d. Installation or integration procedures and limitations, sufficient to ensure that the RSC will be properly used, integrated, and installed. They must be detailed enough to identify unique aspects of the installation or integration. The limitations and procedures must include, as a minimum:

(1) Equipment specifications required for proper operation and performance of the RSC, including verification activities to be performed by the integrator and/or applicant to ensure equipment specifications are met.

(2) A list of any RSC sub-components, as defined by RTCA/DO-178B Section 11.16.

(3) Instructions for periodic maintenance and calibration needed for continued airworthiness once the software is installed on the target environment.

DRAFT

e. Data to support the integrator and/or applicant's completion of partially satisfied or unsatisfied objectives. As an example, if partial credit is sought for objective 1 of RTCA/DO-178B Table A-1 (Software development and integral processes are defined), clearly define to the integrator and/or applicant what that partial credit entails and what they need to do to complete the credit for the installation. The necessary data to support that "full" or "partial" credit must also be made available to the integrator and/or applicant.

f. Software verification results, verification cases, and verification procedures, especially for that verification that must be repeated by the integrator or applicant in the context of the integrated software installed on the target computer environment. Examples of verification to be carried out by the integrator and/or applicant include data coupling analysis, control coupling analysis, timing analysis, memory analysis, software integration testing, hardware-software integration testing, and robustness testing of RSC functions, including safety and protection features. The verification data should include a list of test cases and procedures affected by any settable parameters. The integrator and/or applicant should consider the total requirements for system and sub-system testing; within this context the integrator and/or applicant should address:

(1) applicable credit for reusable tests of the RSC;

(2) re-test where new settings or parameters may affect the requirements, code, function, performance, or protection features;

(3) analyses of data coupling and control coupling of the RSC, including guidance for the integrator or applicant to facilitate the data coupling analysis and control coupling analysis of the RSC integrated with the other airborne software components of their system; and

(4) development of new test cases and procedures to complete all test and test coverage analyses objectives, including guidance for the integrator or applicant to facilitate demonstrating normal range and robustness testing and test coverage objectives for the entire integrated airborne software.

g. Open problem reports on the RSC and analysis of any potential functional, operational, performance, and safety effects. The RSC developer should document this information in the RSC SAS, and if the information is known at the beginning of the project, include it in the RSC PSAC.

h. The RSC developer must develop a data sheet for the RSC. This data sheet must summarize RSC functions, limitations, analysis of potential interface safety concerns, assumptions, configuration, supporting data, open problem reports, software characteristics, and other relevant information in a concise manner that supports the integrator and/or applicant's use of the RSC. The data sheet must be submitted to the FAA and will be attached to the FAA acceptance letter.

DRAFT

i. The following data-related items must also be addressed by the RSC developers (although they may not result in submittals):

(1) Any RTCA/DO-178B software life cycle data not listed above, but used in the software development and approval process, must be made available to the applicant, integrator, and certification authority (for example, Software Quality Assurance (SQA) and Software Configuration Management (SCM) records).

(2) Irrespective of any legal and proprietary issues and agreements about the delivery of software life cycle data between the applicant and the RSC developer, the data must be available to the certification authorities (and authorized designees) at all times for their review and inspection. A process may be set up to make some data available to the applicant without actually supplying the data to the applicant (for example, a data/software escrow). This data must be accessible to the certification authority (and authorized designees) to determine compliance, or in the event of safety or operational problems with the target system (see 14 CFR § 21.277). The data may also need to be available to the applicant, if the target system or RSC requires modification (reference 14 CFR § 21.301 through § 21.305, and FAA Order 8110.4).

(3) Data needed to support changes to the RSC must be identified and maintained. For example, if the developer goes out of business, this data will support continued airworthiness and operational safety. 14 CFR, Part 21, Certification Procedures for Products and Parts (as supplemented by FAA Order 8110.4, Type Certification (Chapters 2 and 3)), provides guidance for the issuance and preservation of type design data for maintaining the continued airworthiness of aircraft products.

(4) The RSC developer must retain and maintain a list of all integrators and applicants buying or using their components to support continued airworthiness across multiple products. The RSC developer and integrators/applicants must set up a process to share in-service problem reports in support of operators required to comply with 14 CFR § 21.3, and in support of Sections 8.n and 8.o below. The RSC developer and user(s) must develop an agreement to support continued airworthiness of the system(s) using the RSC.

8. <u>GUIDELINES FOR THE INTEGRATOR AND APPLICANT USING THE RSC</u>. Sometimes the integrator and applicant are the same company or organization and sometimes they are separate entities. The guidelines for the integrator and applicant are listed below. The applicant is responsible for ensuring that these items are completed, even if some of the tasks are actually performed by an integrator. The applicant and/or integrator must perform the following for each RSC integrated into their system application:

a. Integrate the RSC developer's plans, documentation, limitations, compliance statement, mapping to RTCA/DO-178B objectives, software approval approach, and other relevant information (such as, RSC acceptance letter and data sheet) into their own software life cycle data.

DRAFT

b. Specify the RTCA/DO-178B software life cycle data needed from the RSC developer that supports their project and continued airworthiness. This data is listed in Section 7 of this AC.

c. Produce a system-level PSAC (and/or equivalent certification plan) for the target system, including the information outlined in RTCA/DO-178B, Section 11.1. The system-level PSAC must include the integrator and/or applicant's plans to address compliance with all RTCA/DO-178B objectives, regulations, and guidance for the RSC and other software components of the target system. Additionally, the system-level PSAC must explicitly state the agreement that the RSC was developed with the intent to be reusable in other projects and that they intend to comply with this AC.

d. Produce other system-level software plans (such as, SDP, SCMP, SVP, and SQAP) for their target system. Each plan must address the RSC integration and other software components used. For example, the system-level SVP must cover the overall software verification program, plus any verification required to integrate the RSC and other components, and the credit proposed for the RSC developer's verification.

e. Evaluate the safety, operational, performance, and functional impacts of the issues identified in the RSC developer's PSAC, SAS, and safety analysis data; determine the applicability and severity of these impacts on the specific application and system; determine any additional impacts for the specific application; propose risk mitigation, system architectural design features, protection mechanisms, and other assurance methods to address those risks; and address all safety, operational, functional, and performance issues during the development of the system.

f. Coordinate all plans and standards (as needed) with the certification authority and designees (if delegated) to get agreement on the project.

g. Follow the approved plans and standards. Should any deviations from the plans or standards be necessary, those deviations should be coordinated with the certification authority (and authorized designees) prior to implementation.

h. Analyze open problem reports on the RSC (including development problem reports and in-service problem reports), other software components, hardware, and system to ensure that there are no safety, operational, functional, or performance effects from the RSC or other components in the specific application and system.

i. Validate that the assumptions for RTCA/DO-178B objective credit made by the RSC developer in the RSC SAS are met. Demonstrate the applicability of the credit to the integrated system that uses the RSC and complete the RTCA/DO-178B objectives that were identified as "partial" or "no" credit in the RSC SAS. The applicant is responsible for ensuring compliance to all applicable RTCA/DO-178B objectives for the integrated RSC.

j. Evaluate and address the common reuse issues described in Section 12 of this AC for the each particular application.

DRAFT

k. Validate and verify the throughput, timing, memory usage, resource usage, and other resource items of the RSC and other installed software components for the specific target environment.

l. Keep the certification authority and designees (if applicable) informed of the project status and approved plan deviations. This communication supports timely reviews by the certification authority and/or designees (if applicable) and approval of changed plans.

m. Submit all SCIs, SASs, and other required software life cycle data to the certification authority (that is, submit both system-level and RSC data). The system-level SAS must include the information described in Section 11.20 of RTCA/DO-178B for the system's software. The system-level SCI and SAS must identify that the RSC has been included in the applicant's project, the configuration (including part numbers) of the RSC, the configuration (and part numbers) of other software components, and the software life cycle data configuration to support the RSC and other software components used in the system. Additionally, the system-level SAS must include a description of how RTCA/DO-178B objectives that were not fully met by the RSC developer have been completely satisfied by the integrator and/or applicant for the entire integrated system.

n. Report in-service difficulties with the RSC to the RSC developer and the certification authority who granted the acceptance letter.

o. For subsequent use of the RSC, investigate the in-service experience related to the RSC to ensure that no safety-related problems connected with the RSC have been experienced. Relevant information, such as problem reports available to the RSC developer, must be evaluated for this purpose (see 7.i(4) above). Safety-related in-service experience relative to the RSC must be communicated to the RSC developer and the certification authorities.

p. Establish a legal agreement with the RSC developer regarding continued airworthiness support, data ownership, and so forth, as required to meet the regulations.

9. <u>EXPECTATIONS FROM CERTIFICATION AUTHORITIES ON THE FIRST USE OF AN RSC</u>. The RSC developer, integrator, and applicant should work closely with the certification authority throughout the RSC development and integration. To gain acceptance of an RSC in its first system installation, the certification authority will typically:

a. Coordinate and work closely with the applicant, integrator, and RSC developer to ensure that they comply with the guidance of this AC.

b. Involve directorate personnel, headquarters personnel, technical specialists, and chief scientific and technical advisors (CSTAs), as needed, to address policy and technical issues in the project.

DRAFT

c. Review the RSC developer, applicant, and/or integrator's plans to ensure that the applicable RTCA/DO-178B objectives, regulations, and guidance will be satisfied.

d. Perform on-site or desk reviews of the software life cycle data and the capability of the RSC developer, applicant, and integrator, as needed, to ensure compliance to the applicable RTCA/DO-178B objectives, guidance, and regulations.

e. Ensure that a process is established between the applicant and RSC developer to address any continued airworthiness and in-service problems (see Sections 7.i(4), 8.n, 8.o, and 11.h of this AC).

f. Approve data from the applicant, integrator, and RSC developer (as in a typical software program) for the system software, when the stakeholders satisfactorily complete their development and compliance activities.

10. <u>THE RSC ACCEPTANCE LETTER</u>. If this AC is followed, upon successful certification of the product or authorization of the equipment using the RSC, the certification authority will write an acceptance letter for the RSC and will submit it to the RSC developer. A copy will be provided to the applicant and/or integrator. This letter documents the initial acceptance of the RSC and its suitability for use in other certification projects by other applicants and/or integrators. The acceptance letter typically includes:

a. The RSC document numbers and revision levels approved (for example, the SCI number and revision; the SAS number and revision; and any additional configuration information not included in the SCI), and a general description of the RSC functionality and target environment.

b. The RSC developer's name and contact information.

c. The name and contact information of the original RSC applicant and/or integrator, the airborne system and environment, and other relevant information pertaining to the initial acceptance of the RSC.

d. Assumptions made by the RSC developer during the acceptance, including a reference to the RSC developer's SAS. This must include assumptions for each applicable RTCA/DO-178B objective. The assumptions must be sufficiently detailed that other certification authorities, RSC integrators, and applicants could apply the information to subsequent projects.

e. Summary of technical or policy issues that arose during the initial acceptance and how those issues were addressed.

f. Summary of extra activities performed by the integrator and applicant to assure the RSC for the initial system approval, including system bench and aircraft testing.

DRAFT

g. Contact information for the certification office that will address future questions about the RSC acceptance and subsequent reuse.

h. Software level of the RSC, any RSC limitations, and known installation, safety, operational, functional, or performance issues of the RSC.

i. RSC data sheet, as described in Section 7.h of this AC. A copy of the RSC data sheet should be attached to the acceptance letter.

j. Emphasis that acceptance of the RSC in one project is not approval in any other project. Any subsequent user of the RSC must evaluate installation, safety, operational, functional, and performance aspects of the RSC in their application. Additionally, subsequent users of the RSC must evaluate complete compliance to all applicable RTCA/DO-178B objectives, regulations, and guidance for the RSC and other components in their system.

NOTE: Certification authorities may encourage RSC developers to document some or all of the information listed in 10.a through 10.j in their data sheet and/or SAS. Therefore, the certification authority can simply reference the data sheet and/or SAS in the acceptance letter. In this case, the data sheet and/or SAS number, title, and revision level will be included.

11. <u>EXPECTATIONS FROM CERTIFICATION AUTHORITIES ON THE SUBSEQUENT USE OF AN ACCEPTED RSC</u>. When a previously accepted RSC is submitted for subsequent reuse in another product/project or by another integrator or applicant, the certification authority or the designee (if delegated) will:

a. Review the RSC acceptance letter that documents the initial acceptance. This letter may be obtained from the RSC developer or the certification authority office that originally issued the acceptance.

b. Contact the certification office that performed the first acceptance (as documented in the acceptance letter) to discuss project details and to address any questions or concerns.

c. Coordinate and work closely with the RSC applicant and integrator to ensure that they follow this AC's guidelines, address the common reuse issues in Section 12 of this AC, and address any additional certification issues.

d. Ensure that installation, safety, operational, functional, and performance aspects of the RSC in the specific system have been analyzed and addressed.

e. Involve FAA Directorate personnel, headquarters personnel, technical specialists, and/or CSTAs, as needed, to address policy and technical issues in the project.

f. Review the integrator and/or applicant's plans to ensure: (a) the objectives of RTCA/DO-178B will be satisfied; (b) other applicable regulations, guidance, and

DRAFT

agreements will be satisfied; and (c) the assumptions and requirements documented for the RSC and for other software components used in the target system will be satisfied.

g. Perform, where deemed necessary, on-site and desk reviews of the integrator and/or applicant's data and organization's capability (as needed). This ensures compliance to the applicable RTCA/DO-178B objectives, regulations, guidance, and approved plans. It also ensures compliance with the assumptions and requirements documented for the RSC and other software components.

h. Evaluate the in-service experience related to the RSC, as communicated in 8.n and 8.o above. Safety-related in-service experience and continued airworthiness concerns must be addressed before accepting the new use of the RSC.

i. Accept the applicant and/or integrator's data for the overall system software after they satisfactorily complete the integration and compliance activities.

j. Inform the original certification authority of the subsequent software acceptance, and report any issues that arose during the acceptance.

12. <u>COMMON SOFTWARE REUSE ISSUES</u>. There are several issues that may affect the reuse of software components. Some of the most common issues are discussed below (this is not an exhaustive list):

a. Requirements definition.

(1) RTCA/DO-178B discusses several types of requirements, including system requirements, safety-related requirements, high-level requirements, low-level requirements, and derived requirements. The RTCA/DO-178B discussion and objectives regarding requirements were developed with a traditional federated system in mind. In the traditional case, a single manufacturer is typically responsible for the software development and integration. When RSCs and multiple stakeholders become involved in the software assurance process, determining the levels of requirements may become more difficult. Therefore, satisfying the RTCA/DO-178B objectives related to requirements requires special attention.

(2) Each RSC developer must establish a plan to satisfy the RTCA/DO-178B objectives related to system, high-level, low-level, and derived requirements. It is unlikely that the RSC developer will be able to satisfy RTCA/DO-178B objectives related to traceability to and compliance and consistency with system requirements, nor will they likely be able to "feed back" RSC derived requirements to the systems safety assessment process to ensure there is no impact of design decisions on the system safety. This will likely result in additional effort for the integrator and/or applicant. The RSC developer's means of addressing requirements must be clearly documented in the RSC PSAC and adhered to by the integrator and/or applicant. The integrator/applicant's means of addressing requirements and the system safety assessment must be clearly documented in

DRAFT

the system-level PSAC. Both plans should also be coordinated with the appropriate certification authorities as early in the program as possible.

b. Re-verification.

(1) When an RSC is reused, the question of how much re-verification needs to be performed often arises. Re-verification activities depend on the specific situation (such as, same or different processor, same or different compiler, same or different compiler options, and so forth). The RSC developer should document their overall verification (and re-verification) plans in the RSC PSAC. Additional details should be provided in the RSC SVP; however, the RSC PSAC should have sufficient detail for the certification authority to determine that the approach will address the RTCA/DO-178B verification objectives. The integrator and/or applicant will also need to address verification objectives in the system-level plans. Some examples of verification objectives that cannot typically be satisfied by the RSC developer and must be addressed by the integrator and/or applicant are:

- integration,
- software integration testing,
- hardware-software integration testing,
- requirements-based test coverage,
- timing analysis,
- memory analysis,
- stack analysis,
- data coupling analysis,
- control coupling analysis,
- robustness testing,
- partitioning and other protection mechanisms integrity validation, and
- any installation-specific testing such as system bench testing, aircraft ground and flight testing, and flight test pilot and human factors specialists evaluations of flight deck effects.

(2) Some common re-verification questions to be considered are (not an exhaustive list):

- How much re-verification is required if a different compiler type or optimization is used?
- How much re-verification is required if a different target environment (microprocessor, memory management unit, timers, memory, input/output devices, databuses, etc.) is used?
- How is data coupling and control coupling analyses performed in the new system for the entire application?
- What re-verification or additional verification is required to integrate the RSC with other software components into the overall system?
- How much structural coverage analysis should be repeated if the target system changes?

DRAFT

- How much re-verification is needed for run-time and compiler libraries?
- If a new target environment is used, what kind(s) of resource issues exist (for example, are there non-deterministic, dynamic memory allocation algorithms with the RSC that could create resource (such as, memory and execution time) issues in the new target environment)? If resource issues do exist, how will re-verification be carried out?
- If a new target environment is used and structural coverage changes (that is, there are different unreached code sections), how will the unreached code be addressed? How will it be assured that deactivated code cannot be inadvertently activated in the new system?

c. Interface. The RSC developer must provide interface data. This data must explicitly define what activities are required by the integrator and/or applicant to ensure that the RSC will function according to its requirements. Typical items included in interface data are:

- configuration parameters
- restrictions on tools
- additional verification activities
- memory and timing requirements
- external resources required by the RSC for proper functioning and performance
- definition of the communication mechanisms between the RSC and other software programs and the communication protocols with hardware components
- accessible variables and their characteristics
- variables and data required from the system and their characteristics (for example, inputs to RSC)
- bus and input/output ports and devices
- access mechanisms

d. Partitioning and protection. Although partitioning and protection will most likely be a function at the system level, the RSC itself may require some partitioning and protection. For example, there may be some maintenance code that is at a different software level than the operational flight program for the RSC. In some cases, the RSC might have specific protocols that facilitate protection and partitioning. These should be documented and evaluated by the integrator, applicant, and certification authorities.

e. Data coupling and control coupling analyses. Data coupling and control coupling and the degree of dependency of data and control interchanges between the RSC and other integrated software and hardware components must be carefully addressed to ensure that all potential side effects of data modifications are fully identified and verified. Each side effect should be analyzed to ensure there is no adverse effect on functionality or performance. For example, all modifications of RSC data should only occur at defined interfaces where the data behavior can be fully controlled by the RSC.

DRAFT

f. Using qualified tools. If qualified tools are used to develop and/or verify the RSC, reuse of those tools must be considered during the RSC development and acceptance. RTCA/DO-178B, Section 12.2 provides additional information on the tool qualification process and the supporting documentation.

(1) When qualified tools are used for the development and/or verification of an RSC, the Tool Qualification Plan and the Tool Accomplishment Summary (or PSAC and SAS for verification tools) must document any portions of the tool qualification that are to be completed by the applicant. For example, test procedures and cases might have some target dependencies and additional verification must be performed by the integrator and/or applicant.

NOTE: Some developers have found that packaging the qualification data for each tool helps with reuse. For example, each verification tool used with an RSC might have its own Tool Qualification Plan, Tool Operational Requirements, and Tool Accomplishment Summary.

(2) The following tool qualification data must be provided to the applicant for all tools used in obtaining acceptance of the RSC:

(a) All tool plans,

(b) Tool Operational Requirements, and

(c) The Tool Accomplishment Summary. For some verification tools the Tool Accomplishment summary may be included in the RSC SAS.

(3) All tool data not listed in Section 12.f.(2) of this AC must be available for review by the applicant and certification authority (and authorized designees), as needed, to support continued airworthiness.

g. Deactivated code. Any information about deactivated code and the associated deactivation mechanisms must be identified by the integrator and/or applicant. Since the RSC may have many features to satisfy a broad audience, an approach to tailor the RSC to the specified requirements of an applicant's application is typically needed. This could result in sections of deactivated code that must be addressed as part of the overall software approval process.

h. Traceability. A number of RTCA/DO-178B objectives address the traceability of system requirements, high-level requirements, low-level requirements, derived requirements, code, and test cases and procedures. When multiple stakeholders and multiple components are involved, this traceability becomes more difficult. Traceability must be addressed and maintained between the RSC, the system software, and the system.

i. Robustness. Since the RSC is developed for use in a variety of applications, it must be developed to anticipate out-of-range data or unexpected input; i.e., it must be developed

DRAFT

to be robust. The robustness must be verified through robustness tests during the RSC development and during the integration of the RSC. Stakeholders must document how they plan to address robustness aspects of the RSC.

13. CHANGES TO REUSABLE SOFTWARE COMPONENTS.

a. RSCs will likely change at some point in time. When an RSC is changed, the original reuse status will no longer apply to the changed component (i.e., the acceptance letter cannot be used for the modified RSC). If the stakeholders desire to change a previously accepted RSC, the software component must be modified using the guidelines of this AC (see Sections 13.b through 13.f) and RTCA/DO-178B Section 12.1 and re-accepted as part of an actual project.

b. When an RSC is changed, a change impact analysis must be performed to identify the changed and affected aspects of the software. The change impact analysis should follow a defined process to determine the potential impact of the change on continued operational safety of the aircraft and to determine the impact of the change to the previously accepted RSC. The following items should be addressed by the change impact analysis, as applicable:

(1) Traceability analysis identifies areas that could be affected by the software change. This includes the analysis of affected requirements, design, architecture, code, testing and analyses, as described below:

(a) Requirements and design analysis identifies the software requirements, software architecture, and safety-related software requirements impacted by the change. Additionally, the analysis identifies any additional features and/or functions being implemented in the system, assures that added functions are appropriately verified, and assures that the added functions do not adversely impact existing functions.

(b) Code analysis identifies the software components and interfaces impacted by the change.

(c) Test procedures and cases analysis identifies specific test procedures and cases that will need to be reexecuted to verify the changes and any potential impacts of the changes, identifies and develops new or modified test procedures and cases (for added functionality or previously deficient testing), and assures that there are no adverse effects as a result of the changes. The absence of adverse effects may be verified by conducting regression testing at the appropriate hierarchical levels (such as aircraft flight tests, aircraft ground tests, laboratory system integration tests, simulator tests, bench tests, hardware/software integration tests, software integration tests, and module tests), as appropriate for the software level(s) of the changed software.

(2) Memory margin analysis assures that memory allocation requirements and acceptable margins are maintained.

DRAFT

(3) Timing margin analysis assures that the timing requirements, central processing unit task scheduling requirements, system resource contention characteristics, interface timing requirements, and acceptable timing margins are maintained.

(4) Data flow analysis identifies changes to data flow and coupling between components and assures that there are no adverse impacts.

(5) Control flow analysis identifies changes to the control flow and coupling of components and assures that there are no adverse impacts.

(6) Input/output analysis assures that the change(s) have not adversely impacted the input and output (including bus loading, memory access, and hardware input and output device interfaces) requirements of the product.

(7) Development environment and process analyses identify any change(s), which may adversely impact the software application or product (for example, compiler options or versions and optimization change; linker, assembler, and loader instructions or options change; or software tool change).

(8) Operational characteristics analysis ensures that changes (such as changes to gains, filters, limits, data validation, performance, interrupt and exception handling, and fault mitigation) do not result in adverse effects.

(9) Certification maintenance requirements (CMR) analysis determines whether new or changed CMRs are necessitated by the software change.

(10) Partitioning/protection analysis assures that the changes do not impact any protective mechanisms incorporated in the design.

NOTE: The above list is not all-inclusive and depends on the product for which the modification is being made.

c. The change impact analysis should determine whether the change to the RSC could adversely affect safe operation of the system or product. The following are examples of areas that could have an adverse impact on installation, safety, operations, functionality, or performance:

(1) Safety-related information is changed. For example:

(a) Previous hazards, identified by the system safety assessment, are changed.

(b) Failure condition categories, identified by the system safety assessment, are changed.

(c) Software levels are changed, particularly if the new software level is higher than the previous level.

DRAFT

(d) Safety-related requirements, identified by the system safety assessment, are changed.

(e) Safety margins are reduced.

(2) Changes to operational or procedural characteristics of the aircraft that could adversely affect flight safety. For example:

(a) Aircraft operational or airworthiness characteristics are changed.

(b) Flight crew procedures are changed.

(c) Pilot workload is increased.

(d) Situational awareness, cautions, warnings, and alerts are changed.

(e) Displayed information which is used to make flight decisions is changed.

(f) Assembly and installation requirements are changed.

(g) Equipment interchangeability and/or interoperability with other equipment is changed.

(h) CMRs are changed or added.

(3) New functions or features are added to the existing system functions that could adversely impact flight safety or operations.

(4) Processors, interfaces, and other hardware components or the environment are changed in such a way that safety, operations, functionality, or performance could be adversely affected (see RTCA/DO-178B, Section 12.1.3).

(5) Software life cycle data (requirements, code, and architecture) is significantly changed in such a way that it could adversely affect safety, operations, functionality, or performance. For example:

(a) Changes to software requirements, design, architecture, and code components (especially those affecting safety-related functions, partitioning, redundancy or safety monitors).

(b) Changes to code (source, object, and executable object) components that perform a safety-related function or changes to a component providing input to a component, which performs a safety-related function. (For this AC, a safety-related function is one that could potentially induce or allow a major, hazardous, or catastrophic failure condition to go undetected).

DRAFT

(c) Changes to characteristics of the development environment impacting the executable object code.

(d) Changes to memory allocation requirements so that memory margins are adversely impacted (for example, less than 5 percent margin remaining).

(e) Changes to timing requirements so that timing margins are adversely impacted (for example, margins are unpredictable or less than 10 percent margin remains).

(f) Changes to input/output requirements (such as bus loading) so that input or output performance is adversely impacted (for example, less than 5 percent margin remains).

(g) Data and control coupling characteristics are adversely impacted (for example, to the extent that more than 50 percent of the coverage analysis must be redone).

(h) Changes to interface characteristics.

d. Additionally, the following items should be identified in the change impact analysis:

(1) Updates needed to ensure that the software change(s) is incorporated in the appropriate software life cycle data, including requirements, design, architecture, source and object code, and traceability.

(2) Verification activities needed to verify the changes and that there are no adverse effects on the system. The change impact analysis should cover how changes that could adversely affect safe operation of the system or aircraft will be verified, so the changed and unchanged software will continue to satisfy their requirements for safe operation. These verification activities may include reviews, analyses, regression testing, requirements-based testing, flight testing, and so on, including reevaluation of existing analyses, reexecution of existing tests, and new test procedures and cases (for added functionality or previously deficient testing).

e. When the applicant or integrator makes changes to the RSC without the RSC developer's assistance, that integrator or applicant becomes responsible for satisfying the applicable RTCA/DO-178B objectives for the RSC and all other components of the system.

f. Changes to an RSC as a result of an airworthiness directive (AD) must be coordinated with the RSC developer, users of the RSC, and the appropriate certification authority to determine how the AD applies to other projects. An AD issued on any system containing an RSC may invalidate that RSC as reusable.

14. CONCURRENT USE OF AN RSC.

a. Sometimes an RSC may be developed for use by concurrent projects. The development of the RSC and the multiple applications using the RSC may progress at the same time. In this situation, the RSC developer must create a "Reuse Plan" (or equivalent document) which typically includes:

(1) Known applications and projects that will use the RSC (including the first applicant). Not all future users may be known when the Reuse Plan is written; therefore, the plan should document plans, procedures, and policies for working with the future users and certification authorities.

(2) The schedule for the multiple applications and projects.

(3) A proposed reuse approach, based on this AC's guidance and the specific project needs. The Reuse Plan should thoroughly address this AC. The reuse approach should also propose a way to efficiently utilize FAA and designee resources. For example, shared reviews and review reports may be proposed to optimize resources of FAA and applicants.

(4) A list of all data items (with specific configuration identification) being developed for each user.

(5) A summary of which data items will be the same for all integrators and/or applicants and which data items are user-specific.

(6) An explanation of data items that differ among users (these may not be suitable for reuse).

(7) A list of affected applicants and certification offices. (Note: In some cases the list of applicants may be proprietary data that can only be shared with the certification authority. Therefore, the list of affected applicants may be documented in a separate document to share with certification authorities only.)

(8) A description of how users will be enabled to use the product correctly (for example, a user's guide or interface document).

(9) A description of how the users will be kept up-to-date during the development and deployment of the product. For example, describe how the integrators and/or applicants will be informed of problems found with the RSC, potential safety issues, and other relevant reporting processes.

(10) Statement of the intent to follow this AC.

(11) A description of how potential changes to the RSC will be addressed, as they are required.

b. The Reuse Plan must be coordinated by the RSC developer with all appropriate certification authorities, applicants, and integrators. All stakeholders must agree on the

DRAFT

approach for concurrently using the RSC. Typically, the FAA office that will likely have first approval of a project using the RSC will serve as the focal for the Reuse Plan.

David W. Hempe
Manager, Aircraft Engineering Division
Aircraft Certification Service

DRAFT

APPENDIX 1 – DEFINITIONS OF TERMS

For purposes of this AC, the RTCA/DO-178B Annex B definitions and the following definitions apply:

a. Access mechanism is the manner in which a software component is called upon to perform its intended function. This includes invocation mechanisms and data flow to and from the component. This is typically part of the interface description data.

b. Applicant is the manufacturer seeking certification of the product or authorization of the equipment. The applicant may be applying for a TC, STC, ATC, ASTC, or TSO authorization.

c. Certification Authority is the organization or person responsible within the state or country for the certification of compliance with the requirements. The certification authority is typically the FAA or foreign certification body engineer.

d. Credit is compliance to one or more RTCA/DO-178B objectives supported by RTCA/DO-178B software life cycle data. This compliance is used to show that the certification basis has been met and the equipment may receive a certificate. This AC makes reference to three types of credit:

(1) Full credit – fully meets the RTCA/DO-178B objective and requires no further activity by the applicant and/or integrator.

(2) Partial credit – partially meets the RTCA/DO-178B objective and requires additional activity by the user to complete compliance.

(3) No credit – does not meet the RTCA/DO-178B objective and must be completed by the user for compliance.

e. Designee is a person who is authorized to make compliance findings on behalf of the FAA for the specific project.

f. Documentation configuration is the numbering and revision status used to identify the configuration of documents used in the development process.

g. First use of RSC is the first acceptance of an RSC in a certification project.

h. In-service difficulty is a problem found during in-service experience.

i. In-service experience is experience gained while the RSC is used in a certificated aircraft or engine.

j. In-service problem report is the documentation of an in-service difficulty.

DRAFT

k. Installation procedures are procedures used to install the reusable software component. These might be documented in the porting description data, interface description data, or similar data.

l. Integrator is the manufacturer responsible for integrating the reusable software component into the target computer and system with other software components.

m. Interface description data identifies the interface details of the reusable software component. It is provided by the RSC developer to the integrator and/or applicant. The interface description data should explicitly define what activities are required by the integrator and/or applicant to ensure that the RSC will function in accordance with its approval basis.

n. Maintenance code is code residing in a airborne computer-based system that interfaces with an onboard maintenance computer or computer used by maintenance personnel. The function of this code is usually to report to the maintenance computer any problems detected during normal operations.

o. Porting description data is data that contains assumptions and limitations on the reuse of the component that must be followed by the user, installer, and/or integrator to ensure correct functioning of the component in a new environment.

p. Reusable software component (RSC) is the software, its supporting RTCA/DO-178B software life cycle data, and additional supporting documentation being considered for reuse. The component designated for reuse may be any collection of software, such as, libraries, operating systems, or specific system software functions.

q. RSC developer is the manufacturer of the RSC.

r. RSC user is an integrator and/or applicant who use the RSC.

s. Settable parameters are software component data that are set before execution of the component.

t. Software characteristics include the Executable Object Code size, timing and memory margins, resource limitations, and the means of measuring each characteristic (reference Section 11.20(d) of RTCA/DO-178B).

u. Software component is some part of the airborne system's software. It is usually defined as performing specific functions within the system.

v. Software life cycle data is data produced during the software life cycle to plan, direct, explain, define, record, or provide evidence of successful completion of activities (see RTCA/DO-178B, Section 11.0). Sections 11.1 through 11.20 of RTCA/DO-178B describe different kinds of software life cycle data.

DRAFT

w. Stakeholders are all the persons and groups involved in the development, integration, and acceptance of the RSC. Stakeholders in this AC are the RSC developer, integrator, applicant, and certification authority. One or more manufacturers may assume the roles of the RSC developer, integrator, and applicant.

x. Subsequent use of RSC is the follow-on use of an accepted RSC. That is, it is not the first use of the RSC.

y. Target computer is the physical processor that will execute the program while airborne.

z. Target computer environment is the target computer and all its support hardware and systems needed to function in its actual airborne environment.

aa. Target environment is the same as target computer environment (above).

bb. User – see "RSC user" above.

cc. Variables are named memory locations that contain data that will change during software execution.

DRAFT

APPENDIX 2 - ACRONYMS

The following acronyms are used in this AC:

AC	Advisory Circular
AD	Airworthiness Directive
ARP	Aerospace Recommended Practice
ASTC	Amended Supplemental Type Certificate
ATC	Amended Type Certificate
CFR	Code of Federal Regulations
CMR	Certification Maintenance Requirement
CSTA	Chief Scientific and Technical Advisor
DER	Designated Engineering Representative
FAA	Federal Aviation Administration
PSAC	Plan For Software Aspects Of Certification
RSC	Reusable Software Component
SAS	Software Accomplishment Summary
SCI	Software Configuration Index
SCM	Software Configuration Management
SCMP	Software Configuration Management Plan
SDP	Software Development Plan
SQA	Software Quality Assurance
SQAP	Software Quality Assurance Plan
STC	Supplemental Type Certificate
SVP	Software Verification Plan
TC	Type Certificate
TSO	Technical Standard Order

APPENDIX 3 – SAMPLE FORMAT FOR RSC DEVELOPER'S TABLE

DO-178B Obj #	Objective Description	Credit Sought	Assumption	Means of Compliance for the Objective	Activities Remaining For Integrator/Applicant
1-1	Software development and integral processes activities are defined.	Note 1	Note 2	Note 3	Note 4
1-2	Transition criteria, inter-relationships and sequencing among processes are defined.	Note 1	Note 2	Note 3	Note 4
1-3	Software life cycle environment is defined.	Note 1	Note 2	Note 3	Note 4
1-4	Additional considerations are addressed.	Note 1	Note 2	Note 3	Note 4
1-5	Software development standards are defined.	Note 1	Note 2	Note 3	Note 4
ETC.					

NOTE 1: Include if FULL, PARTIAL, or NO credit is being sought for the RSC. See Section 6.c(1) of this AC.

NOTE 2: List all assumptions made for the credit claim. See Section 6.c(2) of this AC.

NOTE 3: List data that documents the compliance to this objective. See Section 6.c(3) of this AC.

NOTE 4: List the activities remaining for the integrator and/or applicant to complete the objective. This should be in enough detail that the integrator and/or applicant and the certification authority can clearly understand what remains for the overall acceptance of the system using the RSC. See Section 6.c(4) of this AC.

EXAMPLE CONSIDERATIONS FOR EACH DO-178B OBJECTIVE FOR REUSABLE SOFTWARE COMPONENTS

Appendix F

EXAMPLE REUSE CONSIDERATIONS FOR EACH DO-178B OBJECTIVE

This paper summarizes the DO-178B objectives and some potential reuse issues/considerations specific to each objective. The considerations differ for the reusable software component developer, the integrator/applicant, and the certification authority. These issues/considerations are only examples and will need to be addressed on a program-by-program basis. Each program will have its own unique issues/considerations.

Note 1: In the tables below, the acronym RSC is used to abbreviate reusable software component. RSCD stands for RSC developer.

Note 2: For the certification liaison objectives (10-1, 10-2, and 10-3), the considerations vary depending if the RSC is a first-time approval or a follow-on approval.

Obj #	*Objective Description*	*Example Considerations for reusable component developers*	*Example Considerations for Integrators/Applicants*	*Example Considerations for regulators*
1-1	Software development and integral processes activities are defined. 4.1 a, 4.3	• Create PSAC and other planning documents for reusable software components (RSCs) to conform to AC 20-RSC requirements. • Document assumptions, credit, compliances, and remaining activities for each objective.	• Integrate references to RSC plans, SCI, SAS, data sheet, interface data, and other data.into system-level plans. • Coordinate reusable requirements specified in RSC PSAC into system-level plans. • Address how objectives not met or partially met by the RSC will be completed.	• Review the RSC PSAC and plans with the system-level PSAC and plans for consistency, conformance to DO-178B, and conformance with the guidelines of AC 20-RSC. • Provide early agreement on the proposals for reuse specified in RSC developer and integrator/applicant PSACs. • Ensure conformance to the provisions of AC 20-RSC for all planning documents including PSACs. • Ensure that all technical issues not addressed by guidance are coordinated with directorate, headquarters, technical specialists and the CSTA.
1-2	Transition criteria, inter-relationships and sequencing among processes are defined. 4.1b, 4.3	• Since some of the objectives may not be complete for RSC, interface/integration data and the data sheet should specify the transition criteria between the RSC developer and the user. •	• Integrate transition criteria for the acceptance of the components and SW life cycle data from the RSC developer into the planning documents for integrator/applicant.	• Review the integrator/applicants planning documents in conjunction with the RSC developer's interface/integration data to ensure consistency of the transition criteria
1-3	Software life cycle environment is defined. 4.1c	• RSC developer ensures that any life cycle environment needed to complete the objectives or to interface to the RSC is defined in the interface/integration data.	• Include any changes to the RSC developers life cycle environment definition needed to integrate the RSC into the final product.	• Ensure that the integrator/applicant correctly integrated the needed components in their life cycle environment from the requirements of the RSC.
1-4	Additional considerations are addressed. 4.1d	• Any additional activities required to complete satisfaction of requirements for the RSC for additional considerations should be included as part of the SAS, data sheet, and interface/integration data.	• Ensure that any requirements for additional considerations in the SAS, data sheet, and/or interface/integration data are integrated into the planning documents to include the PSAC.	• Review the RSC's SAS, data sheet, and interface/integration data and the integrator/applicant's PSAC to ensure that all additional considerations are acceptable within the framework of existing guidance or agreements.

For Training Only

1-5	Software development standards are defined. 4.1e	• Any SW standards (e.g. calling, naming conventions etc) that must be followed by the applicant/integrator to ensure that the objectives of DO-178 that have been satisfied are not compromised should be included in the interface/integration data.	• Incorporate any required standards from the RSC developer's interface/integration data into existing standards structures.	• Ensure that any requirements for standards adherence in the RSC developer's SAS, data sheet, and/or interface/integration data are incorporated in the planning documents of the integrator/applicant.
1-6	Software plans comply with this document. 4.1f, 4.6	• No unique requirements	• No unique requirements	• Ensure that the planning documents of both the RSC developer and the applicant/integrator meet this objective.
1-7	Software plans are coordinated. 4.1g, 4.6	• No unique requirements	• The integrator/applicant is responsible for ensuring that any coordination between the RSC developer and themselves are coordinated.	• Examine the integrator/applicant plans with the interface data, data sheet, and SAS requirements and ensure that all requirements have been addressed.
2-1	High-level requirements are developed. 5.1.1a	• RSC developer should propose their high level requirements. These may not map to the integrator/applicant's high level requirements.	• The integrator/applicant should define the high level requirements for the overall system. Some type of reference or mapping should be proposed to the component high-level requirements. • If not all of the RSC developer's high-level requirements are used, they should be addressed. One approach would be to consider them as deactivated code.	• Make sure that the RSC developer, applicant, and integrator address the high-level requirements mapping, as addressed in the previous two columns.
2-2	Derived high-level requirements are defined. 5.1.1b	• No unique requirements	• Identify any RSC requirements that do not map to the system-level requirements. • Ensure that derived requirements of RSC are passed to the system safety assessment process.	• Since the RSC developer cannot know which of their requirements are derived requirements, careful evaluation of the integrators/applicants categorization of the RSC requirements will be required to ensure that all derived requirements have been addressed
2-3	Software architecture is developed. 5.2.1a	• Fully define the RSC's architectural requirements for the using system (e.g. timing, protection/partitioning, memory, interrupts etc.).	• Ensure that the RSC developer's architecture is compatible with the system-level architecture	• Ensure that the integrator/applicant has evaluated the architecture for compatibility with the RSC architecture.
2-4	Low-level requirements are developed. 5.2.1a	• No unique requirements	• The integrator/applicant should map the low level requirements to their high and low level requirements to	• Ensure that requirements are developed.

For Training Only

			ensure the completeness of all of the low level requirements.	
2-5	Derived low-level requirements are defined. 5.2.1b	• Identify all derived requirements to be passed up to the applicant/integrator's safety assessment process.	• Address the low-level derived requirements identified by the RSC developer at the project level. • Ensure that derived requirements of RSC are passed to the system safety assessment process.	• Ensure that all the applicant/integrator has addressed all of the low level requirements for the system.
2-6	Source Code is developed. 5.3.1a	• No unique requirements	• The integrator/applicant should ensure that all of the source code is complete.	• Look for evidence that the integrator/applicant has properly accounted for the RSC source code.
2-7	Executable Object Code is produced and integrated in the target computer. 5.4.1a	• The RSC developers should identify the requirements for producing or integrating their object code into a target computer in the interface/integration data.	• The integrator should ensure that their build procedures are compatible with the RSC developer's interface/integration data. • Follow RSC developers assumptions.	• Ensure that the integrator applicant has evaluated the provisions of the interface/integration data dealing with producing object code in the target computer.
3-1	Software high-level requirements comply with system requirements. 6.3.1a	• Will be difficult to know what the system requirements will be. • If reuse is from a previous certified system and the systems requirements are the same for the reuse functionality then previous compliance may be used.	• Responsible to comply with this objective in the context of the system. • RSC developer's high level requirements may not be integrators high level requirements.	• If the integrator/applicant determines that the RSC developer established compliance then the integrator/applicant must justify this determination. • Possible to see requirements implemented that are extraneous to the system. Should be handled as deactivated.
3-2	High-level requirements are accurate and consistent. 6.3.1b	• Clearly define in their verification plan what parts of this objective have been met by their verification process. • Expect Unambiguous, Sufficiently detailed, and 'Not conflict with each other' to be met by RSC developer.	• Expect the accuracy of requirements to be based in part on the system requirements. • Responsible to complete compliance with this objective. • If some or all of RSC developer's high-level requirements are integrators/applicants low-level requirements then RSC developer's data may be used for objective 4-2.	• Expect compliance data to come from both the RSC developer and the integrator/applicant.
3-3	High-level requirements are compatible with target computer. 6.3.1c	• Could provide data that will be used by the integrator/applicant to show compliance. • If reuse from a previously certified system and the systems are	• Responsible for compliance with this objective • If some or all of RSC developer's high-level requirements are integrators/applicants low-level	• If compliance is based on both RSC developer's and integrator/applicant's data then system architecture and target processor need to be considered in evaluating RSC developer's data.

For Training Only

		system and the systems are sufficiently similar then it is possible to partially meet this objective. Note: the system architecture plays a role as well as the actual target processor.	integrators/applicants low-level requirements then RSC developer's data may be used for objective 4-3.	
3-4	High-level requirements are verifiable. 6.3.1d	• Target processor and system architecture play a role in meeting this objective.	• If some or all of RSC developer's high-level requirements are integrators/applicants low-level requirements then RSC developer's data may be used for objective 4-4.	• If this is a simple set of questions that rely on Engineering judgement and the RSC developer did the judging, then must establish that the architecture of the system the RSC developer used is sufficiently close to the real system architecture to yield the correct answer.
3-5	High-level requirements conform to standards. 6.3.1e	• Must conform to the standards established in their plans	• Can use RSC developer's standards for any high-level requirements that match the RSC developer's high-level requirements. • Must conform to standards established in their plans for high-level requirements they develop. • If some or all of RSC developer's high-level requirements are integrators/applicants low-level requirements then RSC developer's data may be used for objective 4-5.	• Can see two standards for high-level requirements.
3-6	High-level requirements are traceable to system requirements. 6.3.1f	• Same as 3-1	• Same as 3-1	• Same as 3-1
3-7	Algorithms are accurate. 6.3.1g	• Possible to define the accuracy and behavior aspects of an algorithm and find compliance to those definitions.	• Must ensure the RSC developer's defined accuracy and behavior match the actual required accuracy and behavior. • If some or all of RSC developer's high-level requirements are integrators/applicants low-level requirements then RSC developer's data may be used for objective 4-7.	• Expect integrator/applicant to justify use of RSC developer's data.
4-1	Low-level requirements comply with high-level requirements.	• Responsible for their low-level requirements complying to their	• Responsible for their low-level requirements complying to their	• Possible to see requirements implemented that are extraneous to the system. Should be

For Training Only

	6.3.2a	high-level requirements	high-level requirements • Must ensure that the RSC developer's high-level requirements mesh into the systems requirements.	handled as deactivated.
4-2	Low-level requirements are accurate and consistent. 6.3.2b	• Responsible for their low-level requirements	• Responsible for their own low-level requirements • Must ensure that RSC developer's low-level requirements do not conflict with their low-level requirements.	• Determine that integrator/applicant was able to ensure no conflict exists between any of the low-level requirements regardless of the origin of the requirement.
4-3	Low-level requirements are compatible with target computer. 6.3.2c	• May supply data for conformance • If software must run on a specific target computer then can do this	• Must comply with this objective for all low-level requirements in the system. • May use data from RSC developer, if can demonstrate that it is pertinent.	• If RSC developer data is used, then need to determine that the context of RSC developer's work is appropriate for the system
4-4	Low-level requirements are verifiable. 6.3.2d	• Responsible for their low-level requirements	• Responsible for their low-level requirements	• Ensure the objective is met
4-5	Low-level requirements conform to standards. 6.3.2e	• Responsible for their low-level requirements conforming to their standards	• Responsible for their low-level requirements	• Will probably see two requirement's standards
4-6	Low-level requirements are traceable to high-level requirements. 6.3.2f	• Responsible for their low-level requirements tracing to their high-level requirements	• Traceability must be done within the context of the system • Any requirements from RSC developer not traced to higher levels of the system must be treated as deactivated	• Expected to see deactivated functionality.
4-7	Algorithms are accurate. 6.3.2g	• Responsible for the algorithms in their low-level requirements	• Responsible for the algorithms in their low-level requirements	• Ensure the objective is met
4-8	Software architecture is compatible with high-level requirements. 6.3.3a	• Supply architectural description of their software to integrator	• Will have to do the work to meet this objective	• Ensure the objective is met
4-9	Software architecture is consistent. 6.3.3b	• Can do this for their part of the architecture	• Must do this for their architecture • Must ensure that RSC developer's work is appropriate and that RSC developer's architecture is	• Ensure the objective is met

For Training Only

			consistent for the entire system	
4-10	Software architecture is compatible with target computer. 6.3.3c	• May supply data for conformance • If software must run on a specific target computer then can do this	• Will have to do the work to meet this objective • May use data from RSC developer, if can show appropriate	• If RSC developer data is used, then need to determine that the context of RSC developer's work is appropriate for the system.
4-11	Software architecture is verifiable. 6.3.3d	• Can do this for their part of the architecture	• Responsible for this system-wide	• Ensure the objective is met
4-12	Software architecture conforms to standards. 6.3.3e	• Can do this for their part of the architecture	• Responsible for their part of the architecture	• Expect two standards for architecture definition
4-13	Software partitioning integrity is confirmed. 6.3.3f	• Many require a partitioning analysis for a partitioned RSC. • Should document assumptions regarding partition (e.g., number and size of partitions).	• In many cases compliance with this objective will be entirely the responsibility of the integrator/applicant. • If the RSC implements some partitioning, the integrator/applicant will verify the partitioning integrity of the integrated RSC.	• Review the partitioning analysis
5-1	Source Code complies with low-level requirements. 6.3.4a	• Responsible for their part	• Responsible for their part	• Ensure the objective is met
5-2	Source Code complies with software architecture. 6.3.4b	• Responsible for their part	• Responsible for their part • Must do this work for the interfaces between the RSC developer's code and the integrator/applicant's code	• Ensure the objective is met
5-3	Source Code is verifiable. 6.3.4c	• Responsible for their part	• Responsible for their part	• Ensure the objective is met
5-4	Source Code conforms to standards. 6.3.4d	• Responsible for their part	• Responsible for their part	• may see two coding standards – on for the RSC and one for the application using the RSC.
5-5	Source Code is traceable to low-level requirements. 6.3.4e	• Responsible for their part	• Responsible for their part	• Ensure the objective is met
5-6	Source Code is accurate and consistent. 6.3.4f	• Responsible for their part	• Responsible for their part	• Ensure the objective is met
5-7	Output of software integration process is complete and correct. 6.3.5		• Most likely that compliance with this objective will be entirely the responsibility of the integrator/applicant	• Ensure the objective is met

For Training Only

6-1	Executable Object Code complies with high-level requirements. 6.4.2.1, 6.4.3		• Most likely that compliance with this objective will be entirely the responsibility of the integrator/applicant	• Ensure the objective is met
6-2	Executable Object Code is robust with high-level requirements. 6.4.2.2, 6.4.3		• Most likely that compliance with this objective will be entirely the responsibility of the integrator/applicant	• Ensure the objective is met
6-3	Executable Object Code complies with low-level requirements. 6.4.2.1, 6.4.3	• Responsible for their part	• Responsible for their part	• Ensure the objective is met
6-4	Executable Object Code is robust with low-level requirements. 6.4.2.2, 6.4.3	• Responsible for their part	• Responsible for their part	• Ensure the objective is met
6-5	Executable Object Code is compatible with target computer. 6.4.3a	• Provide sufficient information, procedures, and computations to allow the integrator/applicant to establish timing and memory margins in the integrated system.	• Provide procedures to evaluate target-specific issues related to the reusable components.. • Provide procedures to integrate timing and memory data from reusable components into system level timing and memory.	• Typically no credit could be granted to the software RSC developer for this objective, since it is target specific.
7-1	Test procedures are correct. 6.3.6b	• Responsible for any tests they perform that will be used by integrator/applicant for compliance	• Responsible for their tests, including tests to integrate the RSC	• Ensure the objective is met
7-2	Test results are correct and discrepancies explained. 6.3.6c Specific to executable object code.	• Responsible for any tests they perform that will be used by integrator/applicant for compliance	• Responsible for their tests • RSC developer's testing must have been done against an executable that will be in the actual airborne executable. Care must be taken that the tools used by the RSC developer to create executable code from source produced the executable that will be airborne.	• There will be configuration control issues with respect to the executable code tested and the executable code that will be airborne

For Training Only

7-3	Test coverage of high-level requirements is achieved. 6.4.4.1	• Can do this to the high-level requirements they defined. May be used by integrator/applicant is those requirements are his high-level requirements.	• Most likely that compliance with this objective will be entirely the responsibility of the integrator/applicant	• Ensure the objective is met
7-4	Test coverage of low-level requirements is achieved. 6.4.4.1	• Responsible for their low-level requirements testing	• Responsible for their tests	• Ensure the objective is met
7-5	Test coverage of software structure (modified condition/decision) is achieved. 6.4.4.2	• May be able to receive credit on the source code aspects of MC/DC but will not be able to get credit for the traceability from object code to source code. • If target processor and compiler are same as applicant's then object code to source code traceability may have meaning	• Evaluate the RSCD's MC/DC approach for the specific application. • Carry out the source-to-object code traceability for the specific target. Must justify using any RSC developer's data • See Objective 7-2 for additional integrator/applicant considerations	• Ensure the objective is met
7-6	Test coverage of software structure (decision coverage) is achieved. 6.4.4.2a, 6.4.4.2b	• Can receive credit for any decision coverage achieved through their requirements based testing	• See Objective 7-2 for additional integrator/applicant considerations	• Ensure the objective is met
7-7	Test coverage of software structure (statement coverage) is achieved. 6.4.4.2a, 6.4.4.2b	• Can receive credit for any statement coverage achieved through their requirements based testing	• See Objective 7-2 for additional integrator/applicant considerations	• Ensure the objective is met
7-8	Test coverage of software structure (data coupling and control coupling) is achieved. 6.4.4.2c	• Can receive credit for any software structure (data coupling and control coupling) coverage achieved.	• Expect integrator/applicant to show coverage of all data and control coupling as related to the interfaces to the RSC developer's sofware. • See Objective 7-2 for additional integrator/applicant considerations	• Be aware that deactivated code may be more likely in reuse situations • Ensure that the data coupling and control coupling analyses for the interface with the RSC is addressed
8-1	Configuration items are identified. 7.2.1	• No unique requirements.	• Ensure that all of the RSC configuration items are incorporated in the configuration identification scheme for the final product.	• Ensure that the integrator/applicant has referenced the proper RSC configuration identification and correct revisions.
8-2	Baselines and traceability are established. 7.2.2	• Ensure that specific baselines and releases are identified for the integrator/applicant. Care should be exercised to ensure that a	• Incorporate the baseline and releases into the product configuration.	• Establish that the RSC baseline has a fully defined approval basis to include open objectives, problem reports, data sheet, and interface/integration data. Ensure that all

For training only

		traceable baseline is established for reusability.		items are consistent for a given reusable baseline.
8-3	Problem reporting, change control, change review, and configuration status accounting are established. 7.2.3, 7.2.4, 7.2.5, 7.2.6	• Provide an interface to enable the tracking of problem reports encountered by the integrator/applicant.	• Develop procedure to accommodate problems on the reusable components. • Evaluate open problem reports and in-service problems of the RSC to ensure that no safety issues are present in the installation.	• Evaluate the traceability of system and integrator/applicant problem reports through to the RSC developer. Track a problem report through both systems. • Ensure that all open problem reports are evaluated for their potential impact on safety.
8-4	Archive, retrieval, and release are established. 7.2.7	• Establish independently controlled archive and retrieval procedures to achieve maximum reusability credit.	• Archive the required portions of the RSC data per DO-178B and AC 20-RSC	• Ensure that the archive, retrieval, and release procedures are consistent with the level of reuse being provided to the RSC developer.
8-5	Software load control is established. 7.2.8	• No unique requirements.	• No unique requirements	• No unique requirements
8-6	Software life cycle environment control is established. 7.2.9	• No unique requirements	• Provide for any additional controls needed as a result of actions and tools required from the RSC.	• Evaluate the integrator/applicant's evaluation of the integration of any RSC tools and procedures.
9-1	Assurance is obtained that software development and integral processes comply with approved software plans and standards. 8.1a	• No unique requirements	• No unique requirements	• Objective compliance will have to be shown at both sites.
9-2	Assurance is obtained that transition criteria for the software life cycle processes are satisfied. 8.1b	• No unique requirements.	• No unique requirements. The transition criteria should already have implemented for the RSC elements.	• Objective compliance will have to be shown at both sites.
9-3	Software conformity review is conducted. 8.1c, 8.3	• The software conformity review will have to accommodate the incompleteness of the data. However, this should be readily specified as part of the SAS and/or data sheet.	• Credit can be taken for the partial software conformity review done by the RSC developer.	• Objective compliance will have to be shown at both sites. The software conformity reviews will have to be evaluated for completeness to ensure that nothing was missed in the overlap between the RSC developer and the integrator/applicant.

For Training Only

10-1 FIRST TIME	Communication and understanding between the applicant and the certification authority is established. 9.0	• Jointly coordinate with the FAA and applicant/integrator. • Consider future applications of the RSC when assigning a software level.	• Facilitate coordination of FAA and RSCD, as appropriate. • Assure that the software level of the RSC is compatible with the system safety analysis for the specific application.	• Provide timely response to RSC plans. • Involve appropriate FAA specialists. • Inform designees of expectations. • Assess overall compliance of the DO-178B • Assure the assumptions are valid and there have been no changes to data. • Assure validity of the SSA.
10-1 FOLLOW ON TIMES	Communication and understanding between the applicant and the certification authority is established. 9.0	• Jointly coordinate with the FAA and applicant/integrator. • Consider future applications of the RSC when assigning a software level.	• Facilitate coordination of FAA and RSCD, as appropriate. • Assure that the software level of the RSC is compatible with the system safety analysis for the specific application.	• Provide timely response to RSC plans. • Involve appropriate FAA specialists. • Inform designees of expectations. • Assess overall compliance of the DO-178B • Assure the assumptions are valid and there have been no changes to data. • Assure validity of the SSA.
10-2 First Time	The means of compliance is proposed and agreement with the Plan for Software Aspects of Certification is obtained. 9.1	• Any changes unique to the RSC should be incorporated in the RSC PSAC. The RSC PSAC should address any additional considerations unique to the RSC such as the interface/integration data and data sheet, any assumptions required to ensure portability of objectives between different instances of reuse, and overall approach for reuse. • PSAC should express the intent to follow AC 20-RSC	• The PSAC should address the issue of previously developed software and the overall approach for certification. • PSAC should express the intent to follow AC 20-RSC	• Structure the approval of the PSACs with two separate letters. One for the RSC developer and one for the integrator applicant. The integrator applicant will get both letters.
10-2 FOLLOW ON TIMES	The means of compliance is proposed and agreement with the Plan for Software Aspects of Certification is obtained. 9.1	• For a given change, the RSC will have to provide any changes to the PSAC to any integrator applicant for which the change is intended. However the approval is required from only one applicant. • PSAC should express the intent to follow AC 20-RSC • It is recommended that the RSC PSAC be updated to address deviations, since it will be reused.	• If the change is previously approved the integrator/applicant is still required to submit a PSAC or other documentation specifying the change and its associated impact. If the RSC DO-178B objectives have been approved for the changes the associated approval documentation should be referenced. • PSAC should express the intent to follow AC 20-RSC	• The FAA should not consider change proposals directly from the RSC developer without an associated STC, TC, TSO change request (either major or minor). The impact of the change should be reviewed on each system incorporating the change even though the DO-178B objectives of the RSC have been approved.
10-3 First Time	Compliance substantiation is provided. 9.2	• The compliance substantiation is provided in accordance with the assumptions specified in the RSC PSAC. The accomplishment summary should provide all the	• The accomplishment summary for the integrator/applicant will need to identify the accomplishment summary and other life cycle data from the RSC developers. Any	• Ensure that the RSC developer documents credit for each DO-178B objective and remaining activities in the SAS. • Ensure that the applicant/integrator shows how they complete objectives where partial

For Training Only

		assumptions needed to ensure that the objectives are transferable to a new use/application without further reevaluation. •	objectives remaining from the RSC should be identified and the associated integrator/applicant data supporting their satisfaction should be specified. DO-178B section 11.f, h, and k will require special treatment for reusability. Reference to the interface/integration data and data sheet will be required	or no credit were claimed by the RSC developer. • Ensure that all DO-178B objectives are met for the application. • Ensure that the integrator/applicant followed the RSC developer data, assumptions, and limitations. If additional limitations were found, those should be addressed.
10-3 Follow ON	Compliance substantiation is provided. 9.2	For a given change, the RSC will have to provide any changes to the Accomplishment Summary and Software Configuration Index to any integrator applicant for which the change is intended. Any data needed to subtantiate an approval will also have to be provided. However the approval is required from only one applicant.	If the change is previously approved the integrator/applicant is still required to submit an Accomplishment Summary and Software Configuration Index and the change's associated impact. If the RSC DO-178B objectives have been approved for the changes the associated approval documentation should be referenced as appropriate.	The FAA should not consider approving the changes directly from the RSC developer without an associated STC, TC, TSO change approval request (either major or minor). The impact of the change should be reviewed on each system incorporating the change even though the DO-178B objectives of the RSC have been approved as part of another project.

For Training Only

SAMPLE DATA SHEET FOR A RESUABLE SOFTWARE COMPONENT

Appendix G

Green Hills • SOFTWARE, INC. •

Safety Critical Products GHS "Customer" Operating System Certification Data Sheet

EXAMPLE

Function	Name	Signature	Date
Software Engineering	x		
CM	y		
SQA	z		
ST&V	u		
Certification	v		

Document Control Information	
Number:	IN-IXX245-0001-CUSTCDS
Date:	

Table of Contents

1 Introduction 3
1.1 Purpose 3
1.2 Scope 3
1.3 Abbreviations 3
1.4 Referenced Documents 4
1.4.1 Internal Documentation 4
1.4.2 External Documentation 4
2 Environment Description 5
3 Timing Performance 6
3.1 Method 6
3.1.1 API Execution Times 6
3.1.2 Partition Switch Execution Time 8
4 Memory Usage 9
4.1 Methods 9
4.2 Memory Usage Results 9
4.2.1 C Library Memory Usage 9
4.2.2 INTEGRITY-178B Memory Usage 9
5 Stack Usage 11
5.1 Method 11
5.2 Stack Usage Results 11
6 Restrictions 12
6.1 INTEGRITY-178B API 12
6.2 Number of AddressSpaces 12
6.3 Number of Tasks per AddressSpace 12
6.4 Number of Kernel Objects per AddressSpace 12
7 Integrator Considerations for Mitigating Partition Breaches 13
7.1 Intra-Partition Considerations 15

Referenced Tables

Table 1. Test Environment 5
Table 2. Timing Analysis for INTEGRITY-178B API Calls 6
Table 3. Partition Switch Timing 8
Table 4. C Library Memory Usage 9
Table 5. INTEGRITY-178B Memory Usage 9
Table 6. AddressSpace Limits 12
Table 7. Partition Breach Considerations 13

EXAMPLE

1 Introduction

1.1 Purpose

This data sheet contains performance results for the Green Hills Software’s (GHS) Safety Critical Products (SCP). The document also lists known limitations.

1.2 Scope

This data sheet is applicable to GHS Safety Critical Products described in this document. The performance measurements are limited to the hardware platform described in Section 2. The performance results are from the software verification results described in Safety Critical Products GHS “Customer” Operating System Software Verification Results [1].

This data sheet includes sections describing:

- Environment.
- Timing, memory usage, and stack usage performance.
- Limitations.
- Integrator considerations for mitigating partition breaches.

EXAMPLE

1.3 Abbreviations

API	Application Programming Interface
ASP	Architecture Support Package
BAT	Block Address Translation
BSP	Board Support Package
ELF	Extensible Linking Format
CM	Configuration Management
DMA	Direct Memory Access
EDAC	Error Detection and Correction
ELF	Executable and Linking Format
GHS	Green Hills Software
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
JTAG	Joint Test Action Group
NA	Not Applicable
OS	Operating System
PPC	PowerPC
SCI	Software Configuration Index
SCP	Safety Critical Products
SQA I	Software Quality Assurance
SEU	Single Event Upset
SRD	Software Requirements Document

ST&V	Software Test and Verification
SVR	Software Verification Results
TBR	Time Base Register
TLB	Translation Lookaside Buffer
usec	microseconds

1.4 Referenced Documents

1.4.1 Internal Documentation

[1] Safety Critical Products GHS “Customer” Operating System Software Verification Results
IN-IXX245-0001-CUSTSVR

[2] Safety Critical Products INTEGRITY-178B Common Kernel Software Requirements Document
IN-NNNNNN-0002-INCSRD

1.4.2 External Documentation

None.

EXAMPLE

2 Environment Description

The results represented in this data sheet are based on the environment defined in Table 1.

Table 1. Test Environment

Item	Description	Configuration Identification
Hardware Platform	“Customer” Platform	DO-IXX245-0001-CUSTtf
Processor	PPC 8245	NA
Clock Speed	200 MHz	NA
Software	SCP INTEGRITY-178B Software Configuration Index for GHS “Customer” Operating System Includes: • C Library • Common Kernel • PowerPC ASP	IN-IXX245-0001-CUSTSCI

SAMPLE

3 Timing Performance

3.1 Method

The INTEGRITY-178B Timing Analysis consists of collecting the following types of data:

1) API execution times.
2) Partition Switch execution time.

3.1.1 API Execution Times

The API execution times are worst-case execution times derived from the INTEGRITY-178B API Test Cases and Procedures. This provides the worst-case blocking terms for invoking the APIs. These numbers may be used by the end user to verify/confirm their system level timing requirements and for determining the worst case partition switch time (see Section 3.1.2). Therefore, timing is only collected for APIs that "trap" into the kernel (i.e., those that are non-preemptable).

The timing data was collected/analyzed using the following process:
A. *<Details Omitted>*.

EXAMPLE

Table 2. Timing Analysis for INTEGRITY-178B API Calls

Section	API Call	Time[2,5] (usec)
Task Synchronization	CreateHighestLockerSemaphore	2.3
	CreateSemaphore	2.5
	GetSemaphoreValue (HighestLocker)	2.2
	GetSemaphoreValue (Counting)	1.9
	ReleaseSemaphore (HighestLocker)	3.1[6]
	ReleaseSemaphore (Counting)	2.5
	TryToObtainSemaphore (HighestLocker)	2.6
	TryToObtainSemaphore (Counting)	2.7
	WaitForSemaphore	3.2[6]
Synchronous Communication	SynchronousSend	2.1[6,7]
	SynchronousReceive	3.9[6,7]
	TrySynchronousSend	2.0[6]
Clocks and Alarms	CreateVirtualClock	2.1
	GetClockAlarm	3.0
	GetClockAlarmOverruns	2.7
	GetClockPermissions	2.2
	GetClockResolution	2.6
	GetClockTime	2.5

Section	API Call	Time[2,5] (usec)
	SetClockAlarm	1.4
	SetClockTime	2.6
	GetClockName	3.6
Memory Management	GetMemoryRegionAddresses	2.0
Task Management	CurrentTask	N/A[1]
	HaltTask	2.1
	RunTask	3.7[6]
	ExitTask	2.5
	YieldTask	2.7
	GetMaximumPriorityAndWeight	2.0
	SetPriorityAndWeight	2.9[6]
	GetPriorityAndWeight	3.2
	GetActivePriority	2.3
	RaiseCurrentTaskPriority	3.3
	LowerCurrentTaskPriority	3.1[6]
	GetTaskStatus	2.8
	Yield	N/A[3]
	Exit	N/A[4]
	GetTaskStackLimits	2.5
	SetTaskStackPointer	2.3
	GetTaskStackPointer	2.0
	SetTaskProgramCounter	2.5
	GetTaskProgramCounter	2.8
	GetTaskRegister	2.6
	SetTaskRegister	2.8
	SetTaskIdentification	2.4
	GetTaskIdentification	2.5
	SetTaskStatusNotificationMask	2.7
	GetTaskStatusNotificationMask	2.7
	ResetActivity	3.3
IO Devices	GetIODeviceOverruns	2.8
	ReadBlockFromIODevice	N/A[8]
	ReadSubBlockFromIODevice	3.0
	ReadIODeviceRegister	2.0
	ReadIODeviceStatus	2.1
	WriteBlockToIODevice	N/A[8]
	WriteSubBlockToIODevice	2.7
	WriteIODeviceRegister	2.4
	WriteIODeviceStatus	3.2

EXAMPLE

[1] *<Notes omitted>*.

3.1.2 Partition Switch Execution Time

The Partition Switch execution time is the worst case time measured in the Partition Switch Timing SVC plus the worst case blocking term for the partition switch to occur. This SVC collected the partition switch execution time using the following process:
A. *<Details omitted>*.

Table 3. Partition Switch Timing

Worst Case Measured Partition Switch Time	Worst Case Measured API/Exception execution time	Total Worst Case Partition Switch Time
1.1us	3.9us (SynchronousReceive API)	5.0 us

EXAMPLE

4 Memory Usage

4.1 Methods

INTEGRITY-178B is a set of libraries that are linked with the application code to obtain an executable image. The data associated with the memory analysis was generated by creating "null" main programs in both kernel and virtual AddressSpaces and linking them into an executable image. The map file was then analyzed to determine the amount of memory used by INTEGRITY-178B.

The C libraries are Executable and Linking Format (ELF) object archives. This means that objects within the archive that are not used by the user code are not included in the final linked executable. In order to perform a full memory analysis, the C Library archives have to be converted to a single linkable object, which preserves all unused objects. The analysis build then uses this newly generated object file to generate the memory analysis information.

4.2 Memory Usage Results

4.2.1 C Library Memory Usage

EXAMPLE

Table 4 shows the total memory usage of the C Libraries.

Table 4. C Library Memory Usage

Memory Type	Usage (Bytes)
Text	6116
Data	776

The C Libraries will use portions of memory allocated to the .stack, .kstack and heap sections. The size of these sections may be defined by the end user. The amount required by the C Libraries is dependent on the end user's usage of the functions within the C Libraries.

4.2.2 INTEGRITY-178B Memory Usage

Table 5 shows the total memory usage of INTEGRITY-178B.

Table 5. INTEGRITY-178B Memory Usage

Memory Type	Usage (Bytes)	
	Kernel	Virtual
Text	37196	3336
Data	18152	1792

The size of the stack, kstack, and heap sections are user-definable and are included in the totals above. The values included in the link maps above represent reasonable values.

EXAMPLE

5 Stack Usage

5.1 Method

Stack Usage is measured when running the INTEGRITY-178B requirements and coverage based tests. For each test, the kernel stack is filled with a “special” bit pattern before the test is executed. Upon test completion, the stack usage is determined by searching the stack for locations matching the bit pattern.

5.2 Stack Usage Results

2480 bytes of the 16k byte stack (~15%) is used.

EXAMPLE

6 Restrictions

6.1 INTEGRITY-178B API

The user is limited to the API defined in Safety Critical Products INTEGRITY-178B Software Requirements Document [2]. All other calls have been removed and will return an error.

6.2 Number of AddressSpaces

Restrictions to the number of AddressSpaces are either processor or debugger limits. Table 6 defines the limits for various processor types.

Table 6. AddressSpace Limits

Processor	Limit Value	Limit Reason
PPC 603r	*<Value Omitted>*	Debugger
PPC 8245	*<Value Omitted>*	Debugger
PPC 755	*<Value Omitted>*	Debugger
PPC 745	*<Value Omitted>*	Debugger
PPC 8250	*<Value Omitted>*	Debugger

EXAMPLE

6.3 Number of Tasks per AddressSpace

No limit from INTEGRITY-178B. Limited only by amount of available memory.

6.4 Number of Kernel Objects per AddressSpace

This limits each virtual AddressSpace to *<Value Omitted>* Objects.

7 Integrator Considerations for Mitigating Partition Breaches

When integrating INTEGRITY-178B into their system, the user needs to assure the integration does not result in potential partition breaches.

Potential causes of partition breaches a user needs to consider can be categorized as follows:

- Breach due to INTEGRITY-178B's partitioning mechanisms not used or bypassed.
- Breach resulting from hardware failure or design.
- Breach (loss of function) due to not accounting for time utilized to handle exceptions.

Table 7 lists potential causes of partition breaches the user may need to consider. The table is based on an analysis of partition integrity performed by GHS. For each potential cause, a description of mitigation considerations is included.

EXAMPLE

Table 7. Partition Breach Considerations

Potential Cause	Mitigation Considerations
Memory Access Protection Not Provided or is Not Utilized	No claims of partition integrity assurance can be made if the memory access protection device of the processor is not used or is bypassed (e.g., global access is defined for all applications). The processor selected by the user must support memory access protection.
Partition Scheduler Not Utilized	Unless other means is provided to protect against loss of function due to Tasks in other partitions, no claim of partition integrity assurance can be made if the partition scheduler is not used.
Value Overwritten by Kernel Function Provided by the User	Kernel Tasks, device drivers, and exception handlers added by a user have full supervisor privileges. An error in a function with supervisor privileges can result in any memory location being overwritten. The user is responsible for process assurance of all of their defined Kernel Tasks.

Potential Cause	Mitigation Considerations
ROM Failure	May need to consider power-up and continuous tests (e.g., CRC or checksums) or program code. Power-up testing of ROM used to contain the program code is provided by GHS and may be invoked by the user. A function is also provided that can be invoked by a user Kernel Task to periodically test program code residing in RAM.
RAM Failure	This could include slow write followed by a read before the write is completed. Power-up testing of RAM used to contain the program code is provided by GHS and may be invoked by the user. Power-up testing of RAM with initialization sections stored in ROM is provided by GHS and may be invoked by the user. Other testing and design considerations are: • Utilize error detection and correction (EDAC) on RAM. • Perform power-up test. • Continuous testing may conflict with cache devices. • Continuous testing may conflict with partition availability properties (cause excessive blocking time).
Cache Failure	Same considerations as RAM, except caches internal to PowerPC do not include EDAC. If a user does not re-initialize memory following certain types of resets (e.g., warm-start), the user may need to assure the cache has been invalidated/flushed prior to accessing the retained memory locations.

EXAMPLE

Potential Cause	Mitigation Considerations
Memory Access Protection Device Failed	Two simultaneous failures required: ▪ Illegal access to memory attempted AND ▪ Memory protection device failed. Note that before the memory protection device fails, all attempted illegal accesses result in the attempting Task to be halted. To decrease exposure time, the user may consider testing the memory protection device at every power-up when the aircraft is on the ground or at some periodic rate.
Partition Jitter Due to Clock Accuracy and Drift	Partition windows may be longer or shorter than defined due to accuracy of the clock driving the high resolution timer (which utilizes the PowerPC Time Base Register). The clock’s accuracy should be included when establishing time budgets for each AddressSpace. The clock may also drift with environmental factors (e.g., temperature, altitude, and power) as well as age over time. These factors may have to be considered when determining worst-case accuracy.
Other Hardware Failures	The user may have failed hardware devices which may cause misleading data values. This includes analog-to-digital converters and serial devices (which can have stuck bits).
Exceptions Occurring	Exceptions are not blocked on a partition basis. User should account for all exceptions (including page faults) and their expected/worst-case execution time and rate. For external interrupts, user may need to include functionality to detect runaway exceptions (i.e., those whose occurrence exceeds a worst case rate).

EXAMPLE

7.1 Intra-Partition Considerations

While the operating system protects partitions from failures in other partitions, a partition is susceptible to failures induced within the partition. The operating system does not prevent interactions between Tasks, Alarms, and other Objects

defined in an AddressSpace, as these interactions are within the partition boundary. Additional considerations during development may be:

- Task not assigned correct priority to compete with other defined Tasks.
- Task’s priority was changed by another Task.
- Task’s status was changed (e.g., Halted) by another Task.
- Time to transfer a message will be utilized during partition of Task assigned to one or both ends of the Connection.
- Tasks that block will block until the Object they are waiting on (e.g., Semaphore, Alarm, Task, IODevice, Connection) is available.
- The Task is non-deterministic (e.g., causes recursion).
- Not accounting for worst-case execution. The program’s execution, cache (and its behavior), Kernel calls, context-switch time, partition jitter, and exceptions should be a consideration when calculating execution time.
- The Task is no longer doing useful work (i.e., is livelocked) or deadlocked.

EXAMPLE

SAMPLE ACCEPTANCE LETTER

Appendix H

Sample Acceptance Letter

U.S. Department
of Transportation
Federal Aviation
Administration

<Directorate>
<name of ACO>
<address of ACO>

<date>

To: <ABC Company and address>
cc: <XYZ Aircraft Company and address>

Subject: Acceptance of ABC's Reusable Software Component, the Y Component, identified in document 12345, revision #.

Project: <project #>

Dear **<addressee>**:

The purpose of this letter is to document the acceptance of ABC's Reusable Software Component (RSC), the Y Component, identified in document 12345, revision #. The Y Component is an operating system that was accepted as part of the XYZ Aircraft Company's 555 type certificate, dated ##/##/##. The partitioned operating system is accepted with the XXX microprocessor. The board support package that serves as the interface between the Y component and the XYZ company's application is not accepted as a reusable component.

The configuration of the Y component and its supporting data is documented in the ABC Software Configuration Index number 12345, revision #. The acceptance of this component was developed in compliance to AC 20-RSC, *Reusable Software Components*. The agreement for the reusable status of the component is documented in XYZ's PSAC number 789-123-003 (revision #) and ABC's PSAC number 22345 (revision #).

The following details are pertinent to the Y component acceptance:

- The assumptions of the Y component acceptance are documented in section 5.3 of the ABC Software Accomplishment Summary 889-123-002 (revision #). Any applicant or integrator desiring to use this acceptance in the future must adhere to these assumptions.
- The limitations of the Y component are documented in section 4 of the attached ABC Data Sheet 989-123-001 (revision #). Any applicant or integrator desiring to use this acceptance in the future must evaluate and address these limitations.
- In addition to the compliance with AC 20-RSC, the component was integrated and tested with the XYZ application, using a combination of system bench and flight tests.

- The Y component is accepted as a Level C component. Any applicant or integrator desiring to use this acceptance in the future must ensure that their safety assessment supports such a software level.
- The vulnerability assessment for the Y component is documented in the ABC Vulnerability Assessment document 999-123-001 (revision #). Any applicant or integrator desiring to use this acceptance in the future must ensure that their safety assessment considers the Vulnerability Assessment and that any claims made in the assessment supports their installation.
- There are a number of open problem reports that where evaluated to have no safety impact on the XYZ application. The problem reports are listed in section 9.2 of the ABC Software Accomplishment Summary 889-123-002 (revision #). Any applicant or integrator desiring to use this acceptance in the future must evaluate each open problem report and its potential impact on their system.
- The Y component had three derived requirements, which are documented in section 7.6 of the ABC Software Accomplishment Summary 889-123-002 (revision #). Any applicant or integrator desiring to use this acceptance in the future must consider each derived requirement in their system safety assessment process.
- The ABC and XYZ companies implemented a process to share problem reports found in-service and elsewhere. ABC will keep a master list of known problems with the Y component. Any applicant or integrator that desires to use the Y component in the future should evaluate all known problems for safety, operation, performance, and functional impact.
- The acceptance of the Y component is for the version described in the Software Configuration Index 12345, revision #. Any changes to the configuration of the Y component are not addressed by this letter and would need to be reassessed on a subsequent certification or authorization program.

The Y component life cycle data and processes were reviewed by a team of FAA engineers and DERs to ensure compliance to RTCA/DO-178B, as described in section 5.3 of the ABC Software Accomplishment Summary 889-123-002 (revision #). The FAA engineers and designees also reviewed XYZ's implementation of the Y component.

It must be stated that the acceptance of the Y component is being granted in the context of the XYZ program. Any other applicants or integrators desiring to use this component must consider the installation, safety, performance, functional, and operational issues of their particular system, as well as the other items mentioned in this letter.

John Doe, of the XXX Aircraft Certification Office, served as the FAA focal point for this acceptance. Any questions may be directed to John or the the XXX Aircraft Certification Office Manager (phone number ###-###-####).

Sincerely,
+++++++
Manager of XXX Aircraft Certification Office

Course Evaluation Form

Appendix I

IVT or Self-Study Video

Evaluation Form

Software Reuse in Airborne Systems

IVT course # 62836; Self-Study Video #25836

We want your candid opinion on the course you just completed. Your feedback will help us to provide the best possible products and services. Please respond to the questions below. If you have completed via IVT, your instructor will prompt you when to enter your answers in your keypad. If you have completed the video option, complete this form manually and return to your ATM. You must complete and return this evaluation form to your ATM in order to get credit for the video option.

A = Highly Satisfactory B = Satisfactory C = Somewhat Satisfactory
D = Not at all Satisfactory E = Not applicable

1. Clarity of objectives	A	B	C	D	E
2. Clarity of instructions	A	B	C	D	E
3. Ease of navigation	A	B	C	D	E
4. Relevance of content to your job	A	B	C	D	E
5. Relevance of exercises to your job	A	B	C	D	E
6. Effectiveness of presentation of content	A	B	C	D	E
7. Quality of feedback	A	B	C	D	E
8. Quality of instructor/student communication	A	B	C	D	E
9. Supervisor support in course completion	A	B	C	D	E
10. Overall quality of the course	A	B	C	D	E

Software Reuse In Airborne Systems
IVT course # 62836; Self-Study Video #25836
October 29-30, 2003

(This page is optional: complete manually)

What information was most useful to you and why?

What information was least useful to you and why?

Additional comments:

If completing this page after participating in the live ATN broadcast, please fax this sheet to the ATN studio at 405-954-0317.

If completing after watching the video, send to your AIR Training Manager (ATM).