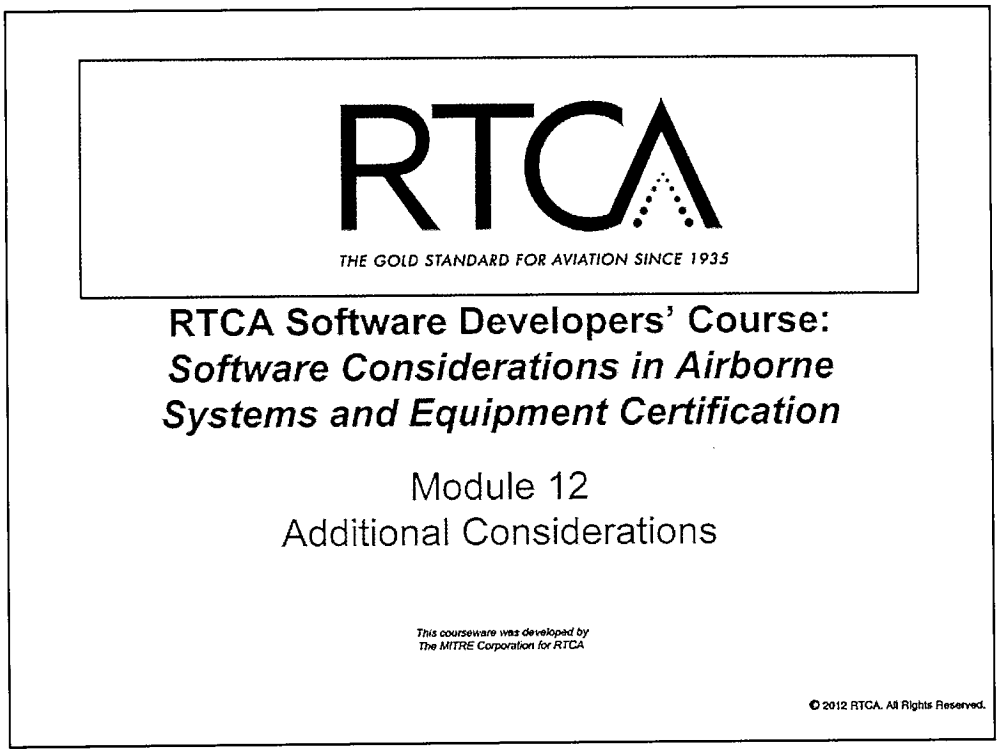
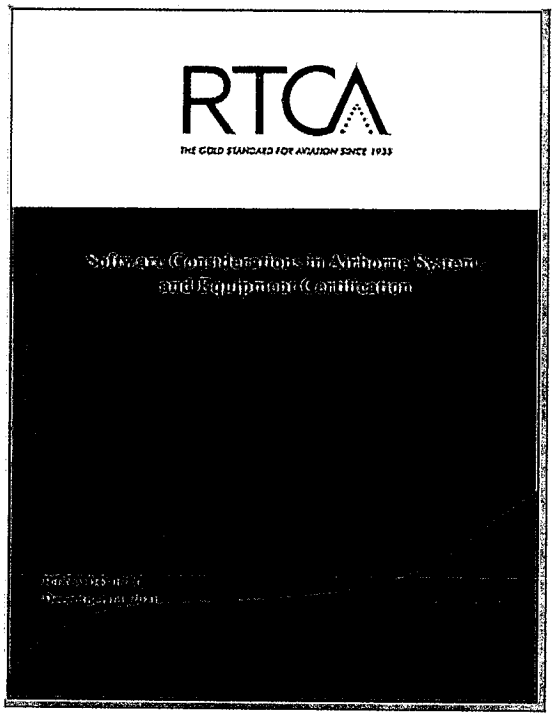


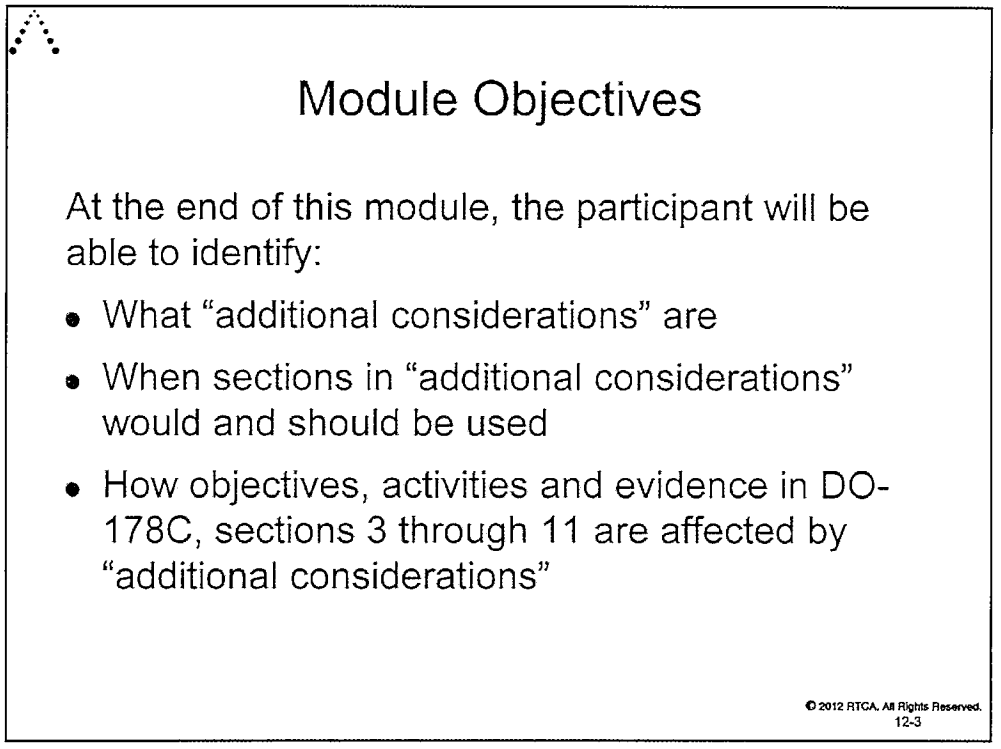
This 3-hour module provides guidance for special conditions that are used as an alternative to the standard considerations used to develop software. This module covers the following topics where objectives and activities may deviate from the objectives and activities defined in DO-178C:

- Previously-developed software
- Development Environment (Tools)
- Exhaustive Input Testing
- Multiple-Version Dissimilar Software Verification
- Software Reliability Models
- Product Service History

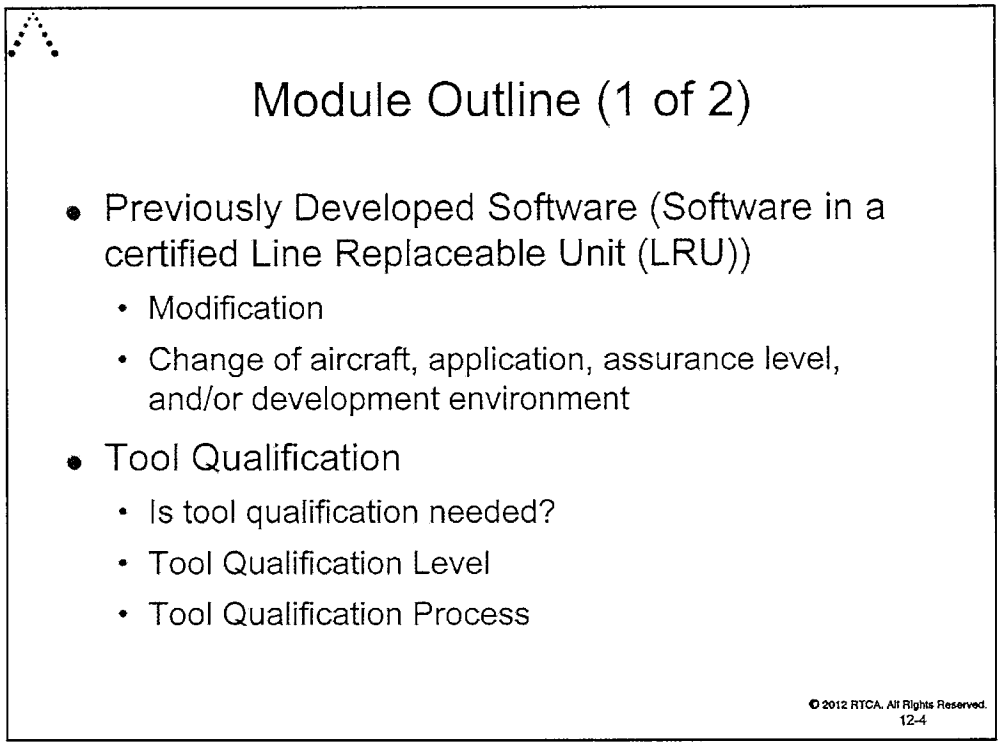
The module identifies when individual sections in "additional considerations" would and should be used, when they would not be used, and how they would affect the objectives, activities, and considerations within DO-178C, sections 3 through 11.



Section 12



This slide covers the objectives of Module 12.



This chart provides the overview of the module.

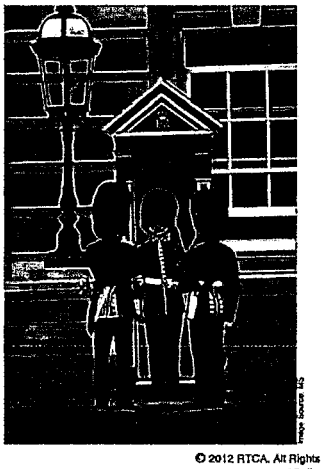
Module Outline (2 of 2)

- Alternate Methods
  - Exhaustive Input Testing
  - Multiple-Version Dissimilar Software Verification
  - Software Reliability Models
  - Product Service History

Previously Developed Software

Topics:

- Modification
- Change of:
  - Aircraft
  - Application or Development Environment
  - Software Level
- Configuration Management
- Software Quality Assurance



Section 12.1

12-6

Modification

This guidance discusses modifications to previously developed software where the outputs of the previous software life cycle processes comply with this document. Modification may result from requirement changes, the detection of errors, and/or software enhancements.

Activities include:

- The revised outputs of the *system* safety assessment process should be reviewed considering the proposed modifications.
- If the software level is revised, the guidance of section 12.1.4 should be considered.
- Both the impact of the software requirements changes and the impact of software architecture changes should be analyzed, including the consequences of software requirement changes upon other requirements and the coupling between several software components that may result in reverification effort involving more than the modified area.
- The area affected by a change should be determined. This may be done by data flow analysis, control flow analysis, timing analysis, traceability analysis, or a *combination of* these analyses.
- Areas affected by the change should be reverified in accordance with section 6.

Previously Developed Software

Topics:

- Modification
- Change of:
  - Aircraft
  - Application or Development Environment
  - Software Level
- Configuration Management
- Software Quality Assurance

Section 12.1

Previously Developed Software Modification

- Activities include:
  - Review revised outputs of *system* safety assessment process considering proposed modifications
  - Consider section 12.1.4 guidance if software level is revised
  - Analyze both impacts of:
    - software requirements changes and
    - software architecture changes, including:
      - Consequences of software requirement changes upon other requirements
      - Coupling between several software components

    that may result in reverification effort involving more than modified area
  - Determine area affected by change using:
    - Data flow analysis, Control flow analysis, Timing analysis, Traceability analysis, or a combination of these analyses
  - Reverify areas affected by change in accordance with section 6

Section 12.1.1

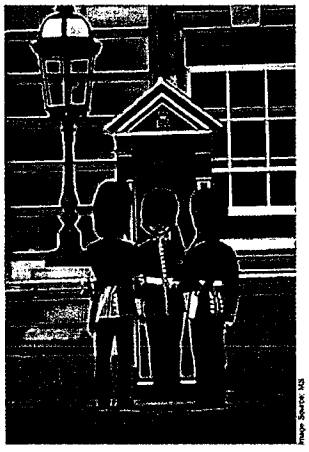
12-8

This chart lists what must be done if there is a modification to Previously Developed Software

Previously Developed Software

Topics:

- Modification
- Change of:
  - Aircraft
  - Application or Development Environment
  - Software Level
- Configuration Management
- Software Quality Assurance



Section 12.1

S

Previously Developed Software Change of Aircraft

- Activities include:
  - The *system* safety assessment process assesses new aircraft installation and determines software level and certification basis
    - No additional effort will be required if these are same for new installation as they were in previous installation
  - Satisfy guidance of section 12.1.1, if functional modifications are required for new installation
  - Satisfy guidance of section 12.1.4, if previous development activity did not produce outputs required to substantiate *system* safety objectives of new installation

Section 12.1.2

This chart lists what must be done if Previously Developed Software is being placed on a different aircraft type

Previously Developed Software

Topics:

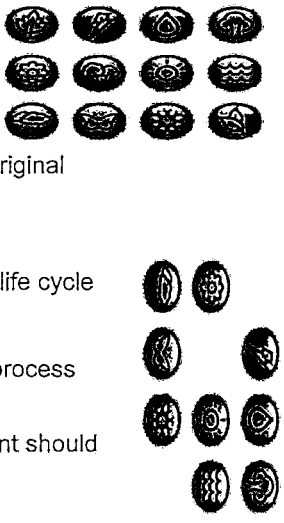
- Modification
- Change of:
  - Aircraft
  - Application or Development Environment
  - Software Level
- Configuration Management
- Software Quality Assurance

Section 12.1

S

Previously Developed Software: Change of Application or Development Environment (1 of 9)

- Changes may involve:
  - New development environment (compiler, etc)
  - New target processor or other hardware or
  - Integration with other software than that used for original application
- New development environments may:
  - increase or reduce some activities within software life cycle
- New application environments may:
  - require *activities*, in addition to software life cycle process activities, that address modifications
- "Changes to an application or development environment should be identified, analyzed, and reverified."



Section 12.1.3

12-12

This chart and the next 5 charts list what must be done if Previously Developed Software is:

- Changed by adding software into the original software
- Used on a different processor
  - If you can no longer buy a particular chip, and you buy a newer model chip, it is "other hardware"; or hardware obsolescence. This may change the software and then you may need to redo all the testing. Would you have to redo structural coverage? Probably not unless the source code changes.
- Uses a different development environment, for example, a different complier
- Used in a different application
  - This is a change of interfaces, requires verification work, and may result in new requirements.

Previously Developed Software: Change of Application or Development Environment (2 of 9)

- Activities include:
  - Apply section 12.2 guidance if new development environment uses software tools
  - Consider complexity and sophistication of programming language to determine rigor of evaluation
    - "For example, rigor of evaluation for Ada generics will be greater if generic parameters are different in new application."
    - "For object-oriented languages, rigor will be greater if objects that are inherited are different in new application."

This is in the Tool Qualification section.

Previously Developed Software: Change of Application or Development Environment (3 of 9)

- Activities include: (continued)
  - "Using different autocode generator or different set of autocode generator options may change Source Code or object code generated."
    - Analyze impact of any changes

Previously Developed Software: Change of Application or Development Environment (4 of 9)

- Activities include: (continued)
  - "If different compiler or different set of compiler options are used, resulting in different object code, results from previous software verification process activity using object code may not be valid and should not be used for new application."
    - "In this case, previous test results may no longer be valid for structural coverage criteria of new application."
    - "Similarly, compiler assumptions about optimization may not be valid."

Previously Developed Software: Change of Application or Development Environment (5 of 9)

- Activities include: (continued)
  - "If different processor is used, then change impact analysis is performed to determine:
    - Software components that are new or will need to be modified as result of changing processor, including any modification for hardware/software integration.
    - Results from previous software verification process activity directed at hardware/software interface that may be used for new application.
    - Previous hardware/software integration tests that should be executed for new application.
      - It is expected there will always be minimal set of tests to be run.
    - Additional hardware/software integration tests and reviews that may be necessary."

Section 12.1.3

Previously Developed Software: Change of Application or Development Environment (6 of 9)

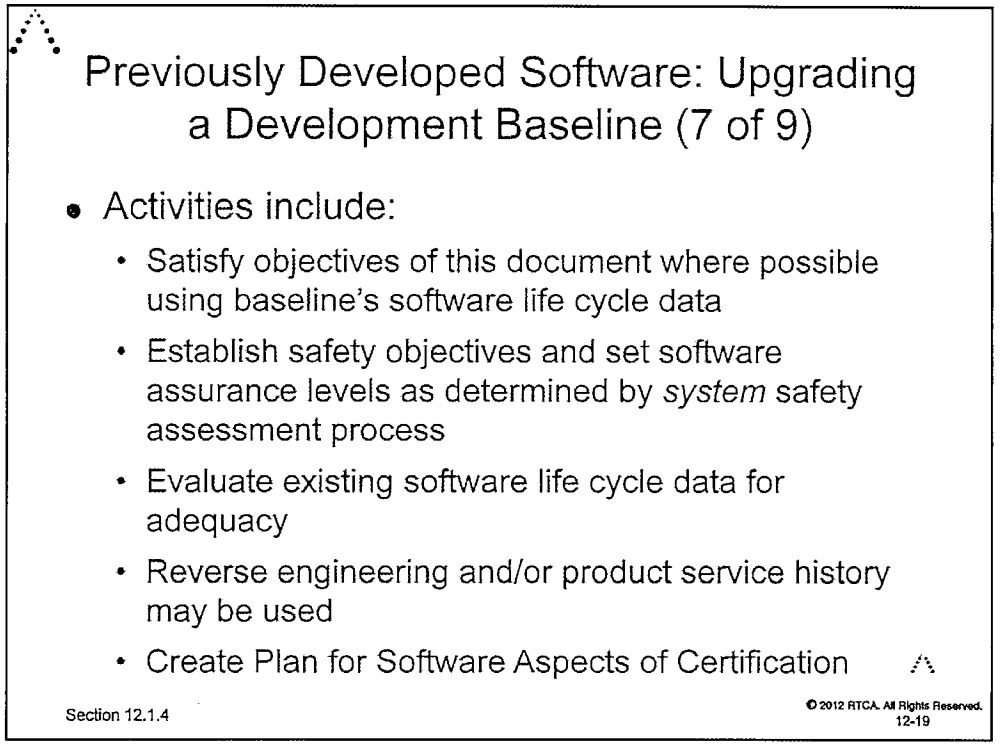
- Activities include: (continued)
  - Perform a change impact analysis if a hardware item, other than processor, is changed and design of software isolates interfacing modules from other modules to:
    - "Determine software modules or interfaces that are new or will be modified to accommodate changed hardware component."
    - "Determine extent of reverification required."
  - Conduct verification of software interfaces where previously developed software is used with different interfacing software
    - Determine extent of reverification required using change impact analysis

Previously Developed Software

Topics:

- Modification
- Change of:
  - Aircraft
  - Application or Development Environment
  - Software Level
- Configuration Management
- Software Quality Assurance

Section 12.1



Applies to:

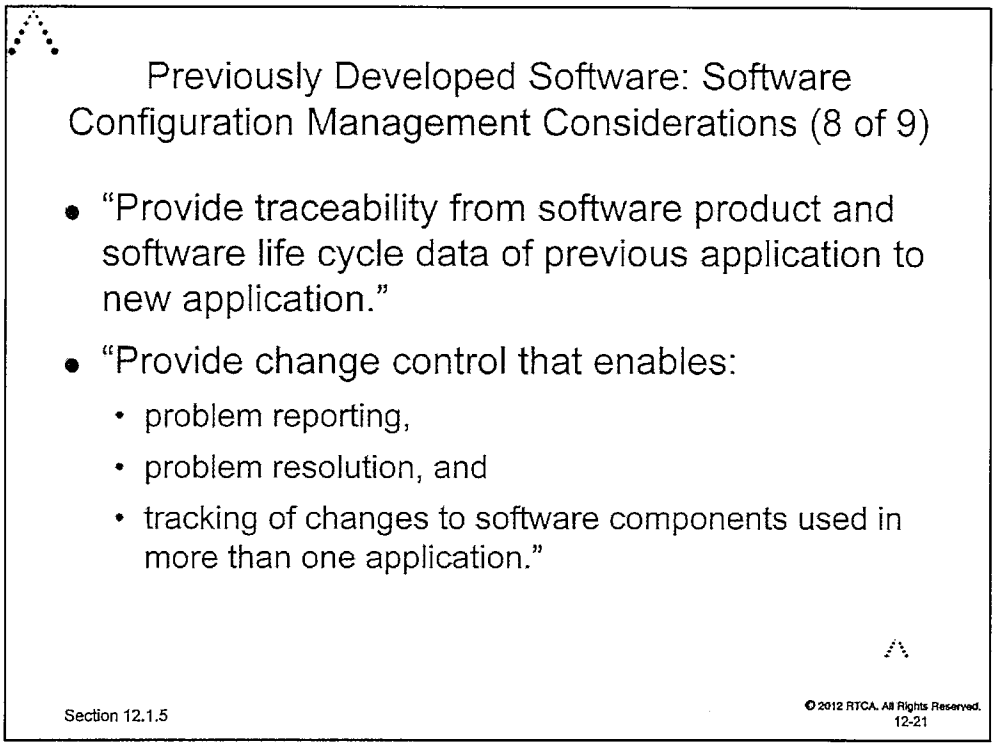
- COTS software.
- Airborne software developed to other guidance.
- Airborne software developed prior to the existence of this document.
- Software previously developed to this document at a lower software level.

Previously Developed Software

Topics:

- Modification
- Change of:
  - Aircraft
  - Application or Development Environment
  - Software Level
- Configuration Management
- Software Quality Assurance

Section 12.1



In addition to all the configuration management objectives and activities of DO-178C, these provide two more possible objectives or activities.

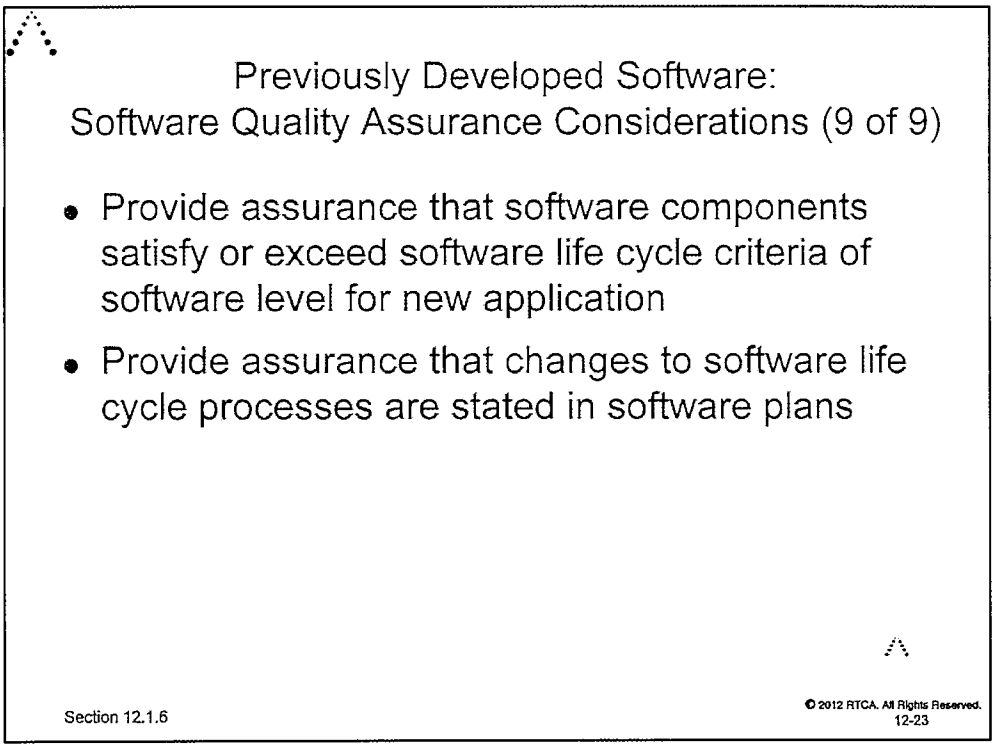
Previously Developed Software

Topics:

- Modification
- Change of:
  - Aircraft
  - Application or Development Environment
  - Software Level
- Configuration Management
- Software Quality Assurance

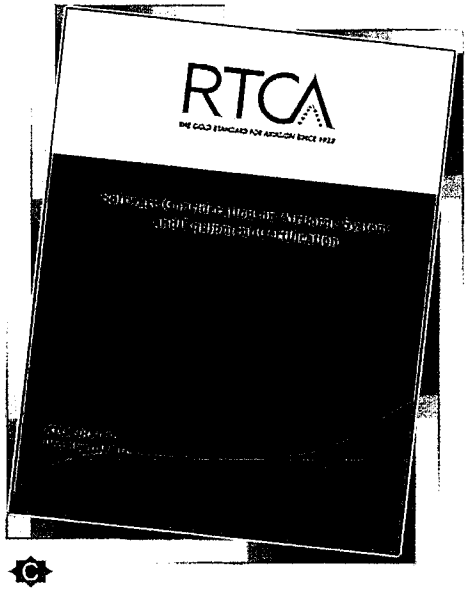
Section 12.1

S



In addition to all the software quality assurance objectives and activities of DO-178C, these provide two more possible objectives or activities.

Tool Qualification



DO-330

"Software Tool Qualification Considerations"

Three Key Tool Qualification Topics

- When is tool qualification needed?
- How do you determine a tool qualification level?
- What is the tool qualification process?

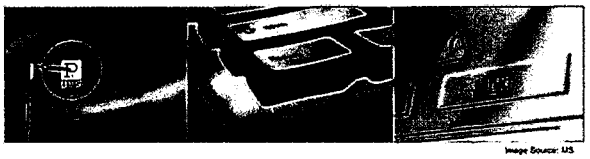
Section 12.2

12-25

This section of the module examines three key topics to consider in Tool Qualification.

Tool Qualification

- Needed when DO-178C processes are:
  - Eliminated
  - Reduced or (code review)
  - Automated



by use of a software tool *without its output being verified* as specified in section 6

Tool qualification – The process necessary to obtain certification credit for a software tool within the context of a specific airborne *system*.

Section 12.2.1

12-26

When considering tool qualification of a tool the first questions to ask are:

- Automation:
  - Are the outputs of this tool any of the Software Life Cycle Data?
  - Will the outputs that are Software Life Cycle Data be verified by a verification activity?

  If the answers are no to the first question, then tool qualification is not required for this tool.
  If the answer to the first question is yes and the answer to the second question is yes, then tool qualification is not required for this tool.
  Example tool: Compiler

Elimination

  - Are the outputs of the processes that this tool eliminates any of the Software Life Cycle Data?
  - Will the outputs that are Software Life Cycle Data be verified by a verification activity?

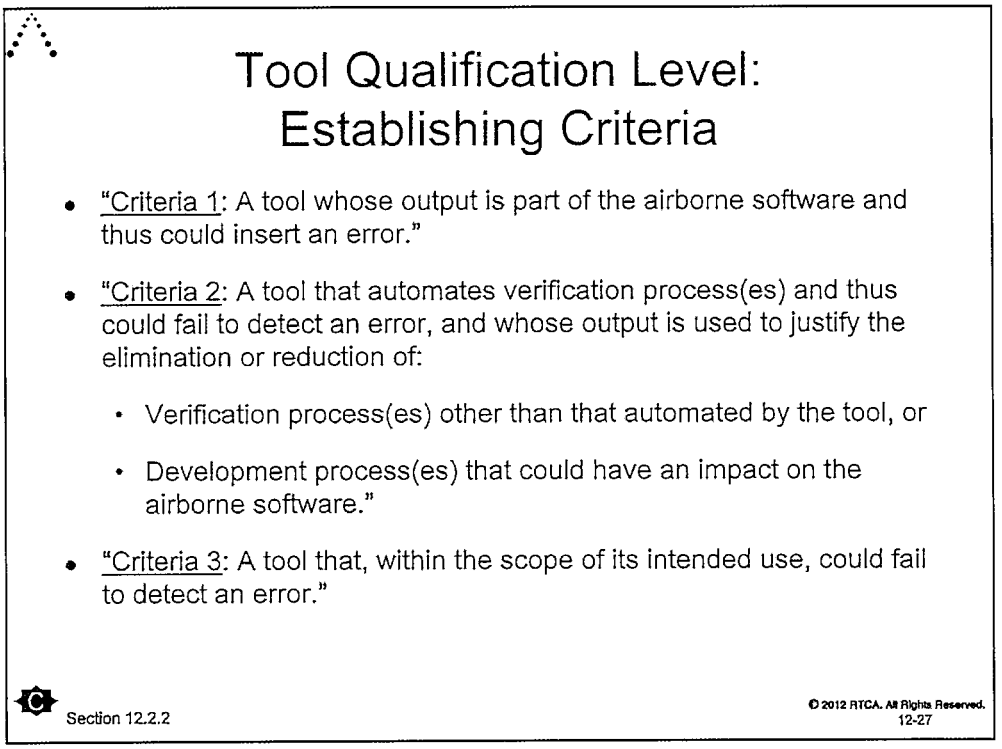
  If the answers are no to the first question, then tool qualification is not required for this tool.
  If the answer to the first question is yes and the answer to the second question is yes, then tool qualification is not required for this tool.
  Example tool : Auto-code generator (Note: Eliminates the Source Code review process)

Reduction

  - Is an activity of a Software life cycle process eliminated (i.e. the process reduced) ?
  - Will the outputs of the Software Life Cycle process be verified by a verification activity?

  If the answers are no to the first question, then tool qualification is not required for this tool.
  If the answer to the first question is yes and the answer to the second question is yes, then tool qualification is not required for this tool.
  Example tool : Smart Editor that ensures the coding standards are met.

A question for the participant – Does a Configuration Management *System* doing version control need tool qualification?



Once it is determined that a tool needs qualification then the tool qualification level (this slide and the next slide) needs to be determined.

The rigor of the tool qualification effort is greatest for Criteria 1 and least for Criteria 3

Criteria 1 means than the tool can change the airborne executable object code without that change being known by the developer . Some examples are a complier, assembler, or auto code generator.

Criteria 2 examples:

> "A *static code analyzer* may be used to automate some verification of Source Code review. Criteria 3 could be applied based on this tool's use and credit claimed. However, if the applicant claims to not include some specific mechanisms in the resulting software in order to detect and treat the possible overflow, and run-time errors based on the confidence on the tool, then the criteria 2 becomes applicable. In this case, it corresponds to "a reduction of software development process(es).""
> DO-330 FAQ D.5

Criteria 3 examples: within the scope of the tool's use, the tool fails to detect an error, i.e., Static Code Analyzer or Simulator.

Tool Qualification: Determining Level

Software Level	Criteria		
	1	2	3
A	TQL-1	TQL-4	TQL-5
B	TQL-2	TQL-4	TQL-5
C	TQL-3	TQL-5	TQL-5
D	TQL-4	TQL-5	TQL-5

Table 12-1

12-28

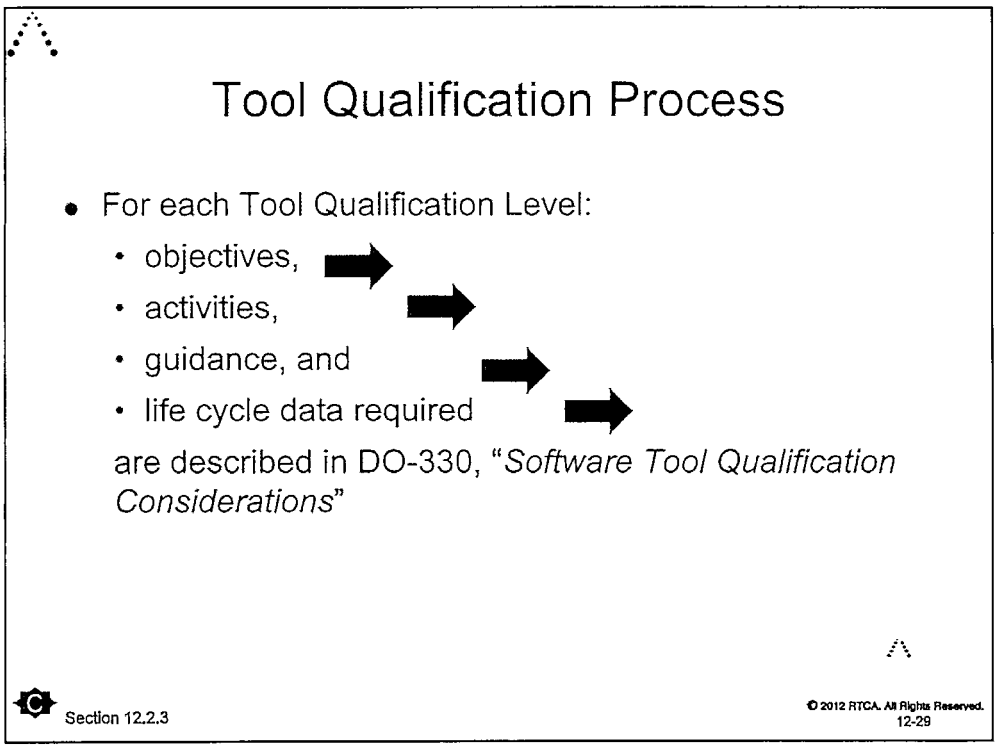
The Tool Qualification Supplement uses 5 levels (like DO-178C) to determine the amount of rigor and evidence needed to qualify a tool. This figure specifies a TQL-1 to TQL-5 (Tool Qualification Level) based on:

- The Software Level of the airborne software being developed
- The Criteria as defined on the previous chart

For Example if the tool is being used to develop airborne software to Software Level B and the Criteria is determined to be 1 (i.e. the tool can introduce an error into the airborne software) then the Tool Qualification Level is TQL-2. The Tool Qualification supplement would then be used to determine what must be done to qualify this tool (see next chart "Tool Qualification Process".)

For Level A software and a Criteria 3 Tool, you will do TQL-5. You take that to the Supplement and use it. If you verify the output of the tool, you don't have to qualify it.

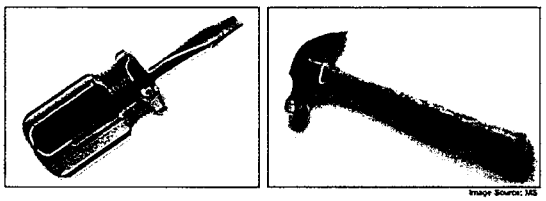
For most of your Criteria 1 Tools, you might consider verifying the output of those tool *instead* of tool qualification.



The Tool Qualification Supplement is very similar to DO-178C. Each TQL number is used to determine which objective, ... are necessary for qualification to that level. Note a TQL-2 tool can be used if only a TQL-4 tool is required but not the reverse.

It is for use in qualifying tools and not to be used to replace DO-178C for airborne software.

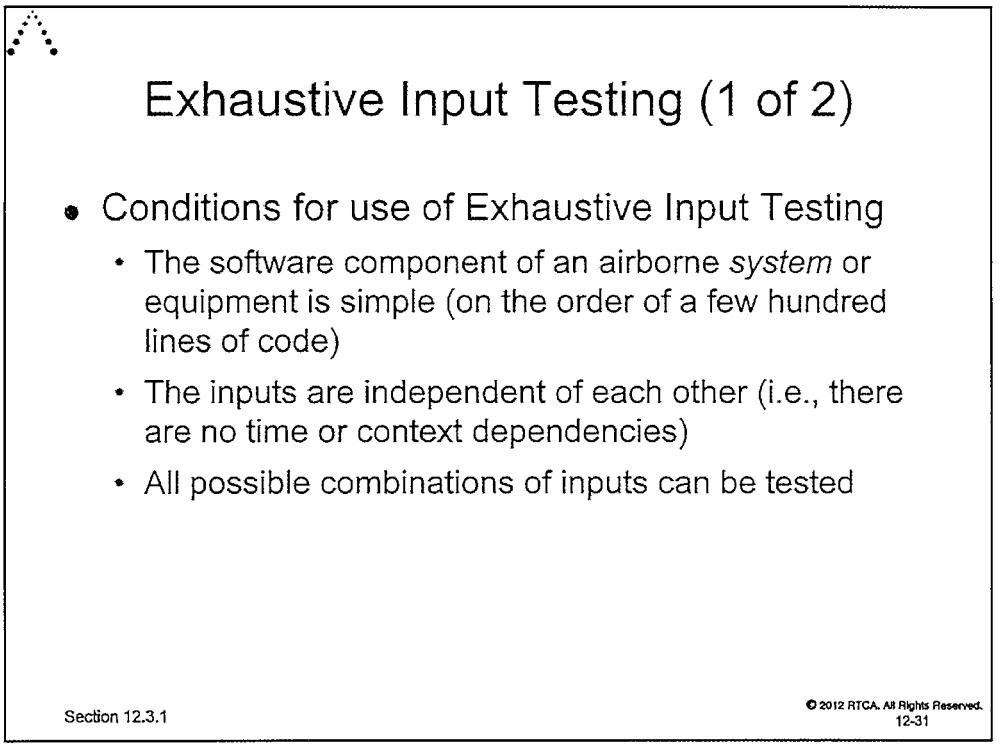
Alternate Methods



- Exhaustive Input Testing
- Multiple-Version Dissimilar Software Verification
- Software Reliability Models
- Product Service History

Section 12.3

We will cover the four alternate methods described on DO-178C in the next set of charts.



Exhaustive Input Testing should not be tried on a large *system* or a *system* with many inputs.

For large *systems* the state space may be too large to allow this method.

For *systems* that have time or context dependencies for the input, this method probably would not be appropriate

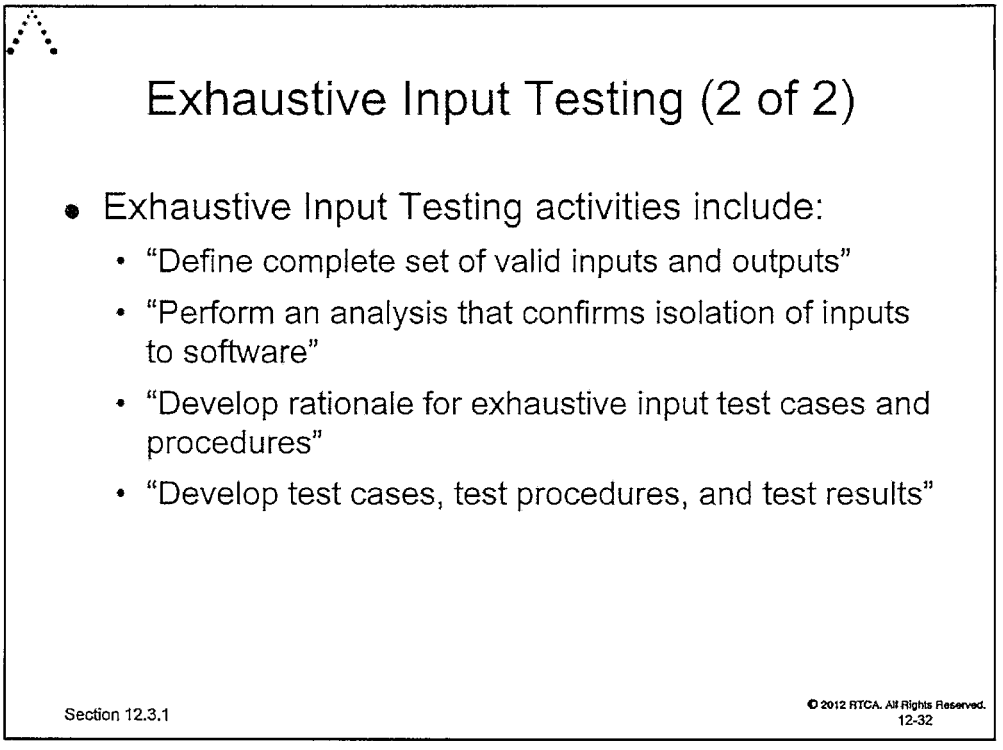
For many inputs the "combination of all possible inputs" may grow so rapidly that the number of test would take months or years to run.

Verification sub-processes that might be eliminated include:

- Structural coverage analysis
- Code reviews
- Traceability

The biggest problem: Show that you are doing exhaustive testing.

Structural Coverage Analysis may not be needed if exhaustive testing is achieved.



Testing must be done. Evidence that all possible combination of inputs were tested must exist and justification that the tests' execution achieved full coverage must be made.

Multiple-Version Dissimilar Software Verification (1 of 2)

- Section 12.3.2 does not enumerate all items that must be considered in order to comply with this alternate method
- When multiple-version dissimilar software is used, single version software methods may be modified

  Examples:
  - Tool qualification
  - Evaluation of any additional code (generated by a compiler, linker, or other means) that is not directly traceable to Source Code

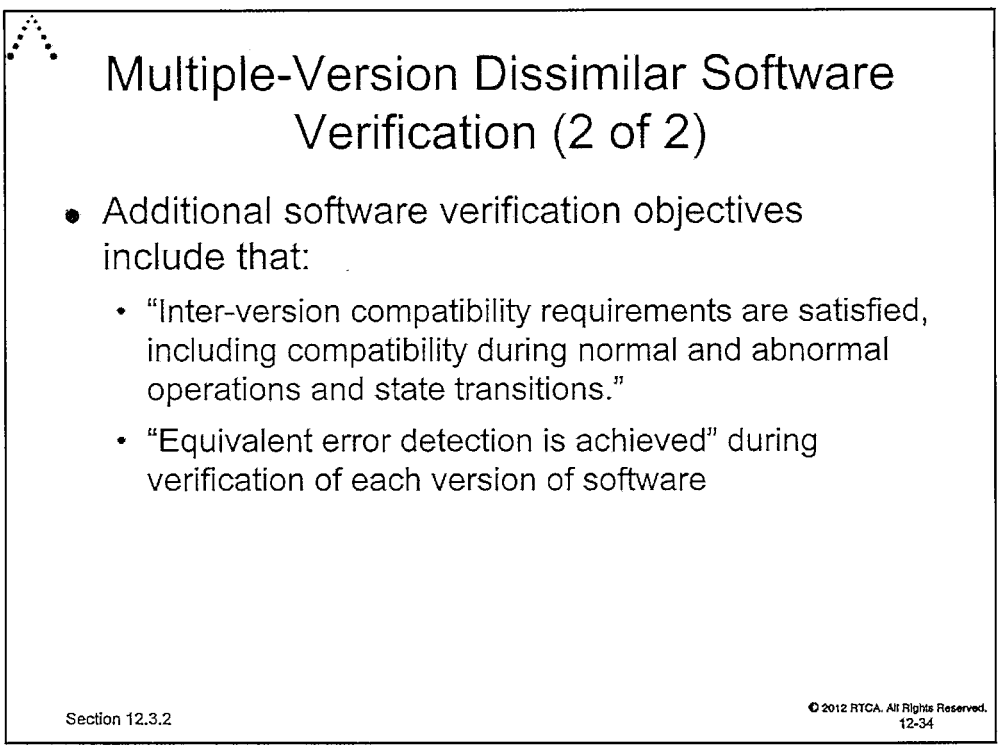
Section 12.3.2

12-33

Achieving dissimilarity is difficult.

For example, it is hard to write dissimilar requirements that yield the same functionality. If the requirements are the same then any errors in the requirements will appear in both dissimilar *systems*.

Dissimilar – How dissimilar? Requirements? Reviews? Testing?



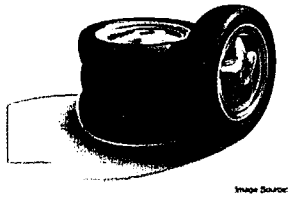
Dissimilar *system* always have a common point were one of the *systems* outputs is chosen over the other (voting). This implies that the output of each must be delivered to the common point with the same meaning of the output. For example, if both *systems* get a bad input, the resulting output must deal with the bad input in different ways but give the same output.

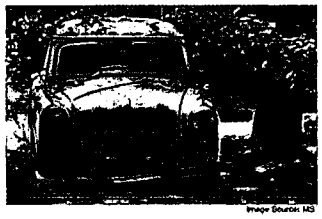
The only advantage is that Independence is eliminated for the Objectives. May not be worth the effort in some cases.

Some Common Mode Failures may be eliminated.

Software Reliability Models

- "Many methods for predicting software reliability based on developmental metrics have been published, for example, software structure, defect detection rate, etc. This document does not provide guidance for those types of methods, because at the time of writing, currently available methods did not provide results in which confidence can be placed."
- Software does not wear-out; Hardware can wear-out
- Software and Hardware can contain "errors" due to incorrect logic





Section 12.3.3

- The probabilities of a software failure are in the probability of a certain set of input values at a specific time.
- Software has a failure class called a design fault. The wrong set of instructions introduced a design error.
- If you write a failure into the software, it will fail every time.
- Why do software failures appear to be random during the software's use? Failures because of deign errors appear to be random failures because the input space has random characteristics.

Product Service History (1 of 5)

- Acceptability of an airborne *system* and equipment's product service history depends on:

• Configuration management of software	• Length of product service history
• Effectiveness of problem reporting activity	• Actual error rates in product service history
• Stability and maturity of software	• Impact of modifications
• Relevance of product service history environment	• Type of problems occurring

- Define and submit to appropriate certification authority:
  - Use of the airborne *system* and equipment
  - Conditions of its use
  - Software service history results from its use

Section 12.3.4

12-36

Use of Product Service History as an alternate method for airborne *system* and equipment depends on:

- The use is for similar purposes
- The environment must be similar
- The adequacy of the methods to document any failures in the field
- The length of service
- The type of problems that were documented
- The modification (type and timing of) made during service

Is the intended use in the correct environment. Products that have been in use for a long time. The problems with Service History are vast; you have to make judgment calls.

Ask your self, how much service history do you need? The applicant has to develop this information and get agreement from the Certification Authority

Level A – very long

Level C – may be able to achieve.

Product Service History (2 of 5)
Relevance of Service History

- Guidance for determining applicability of service history and length of service history needed are:
  - Type of service history
  - Known configuration
  - Operating time collection process
  - Changes to the software
  - Usage and Environment
  - Deactivated code

Section 12.3.4.1

12-37

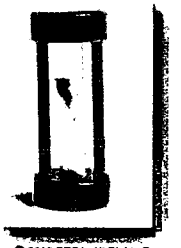
If credit is desired for service history, during the airborne *system* and equipment's service, care must be taken to document all aspects of:

- Use
- Duration
- Configuration

for each installation of the *system* and equipment

Product Service History (3 of 5)
Sufficiency of Accumulated Service History

- *"System* safety objectives of the software and the software level."
- "Any differences in service history environment and *system* operational environment."
- "The objectives from sections 4 to 9 being addressed by service history."
- "Evidence, in addition to service history, addressing those objectives."



Section 12.3.4.2

12-38

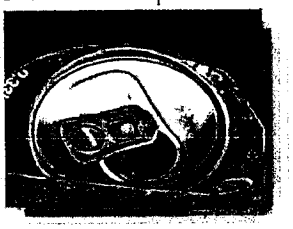
Equipment used with different safety objectives (for example different Software levels) and/or used in different environments contributes only to it's *own* length of service.

The required length of service history will depend on the objectives in DO-178C, sections 4-9, for which service history credit is being sought.

Evidence that there was partial compliance to a DO-178C objective affects the length of service history.

Product Service History (4 of 5)
Collection, Reporting, and Analysis of Problems

- Problem reporting process
  - Agree on specific error data with the certification authority
  - Evaluate chronological trends of the specific error data and explain any increasing trends
  - Address the completeness of the software's specific error data
- Process-related problems
- Safety-related problems



Section 12.3.4.3

12-39

The problem reporting process during the service history must be rigorous. It must be agreeable to the certification authorities based on the criteria listed below. Additionally, metrics (especially trends) must be available from the problem reporting process.

The problem reporting *system* should capture the following about every failure:

1. The hardware/software configuration in effect when the problem occurred.
2. The operating environment within which the problem occurred.
3. The operating mode or state within which the problem occurred.
4. Any application-specific information needed for problem assessment.
5. Classification of the problem with respect to severity, safety significance, and whether the problem was the result of a change in the software configuration since the start of service history data collection.
6. Assessment of whether the problem was:
   - Reproducible.
   - Recoverable.
   - Related to other previously reported problems, including, but not limited to, a common cause.

Process related problems include design or coding errors and must be flagged as such.

All problems evaluated as having a safety impact must be flagged as a Safety-related problem.

Product Service History (5 of 5)

Information Needed when Seeking Certification Credit

- Rationale addressing items in section 12.3.4.1
- Amount of service history needed and rationale
- Rationale for calculating total relevant service history period
- Definition of counted errors and rationale
- Proposed acceptable error rates
- Criteria for problems that would invalidate service history
- Correctable errors:
  - Criteria for defects that will be corrected
  - How they will be corrected and verified
  - Rationale for any defects for which no action will be taken

- A list of objectives in sections 4 to 9 that are replaced by service history

Section 12.3.4.4

This is information that must be supplied to the certification authority when seeking credit for service history.

You show compliance to some section 4-9 objectives by reverse engineering, service history or exhaustive testing.

Some errors in Service History can be corrected.

Review Questions

- What are "additional considerations"?
- When *would*, and *how should*, sections in "additional considerations" be used?
- How do "additional considerations" affect the objectives, activities and evidence within DO-178C, sections 3 through 11?

12-41

Additional considerations are where objectives and/or activities may replace, modify, or add to some or all of the objectives and/or activities defined in DO-178C.

The use of additional considerations should be agreed to on a case-by-case basis with the certification authorities.

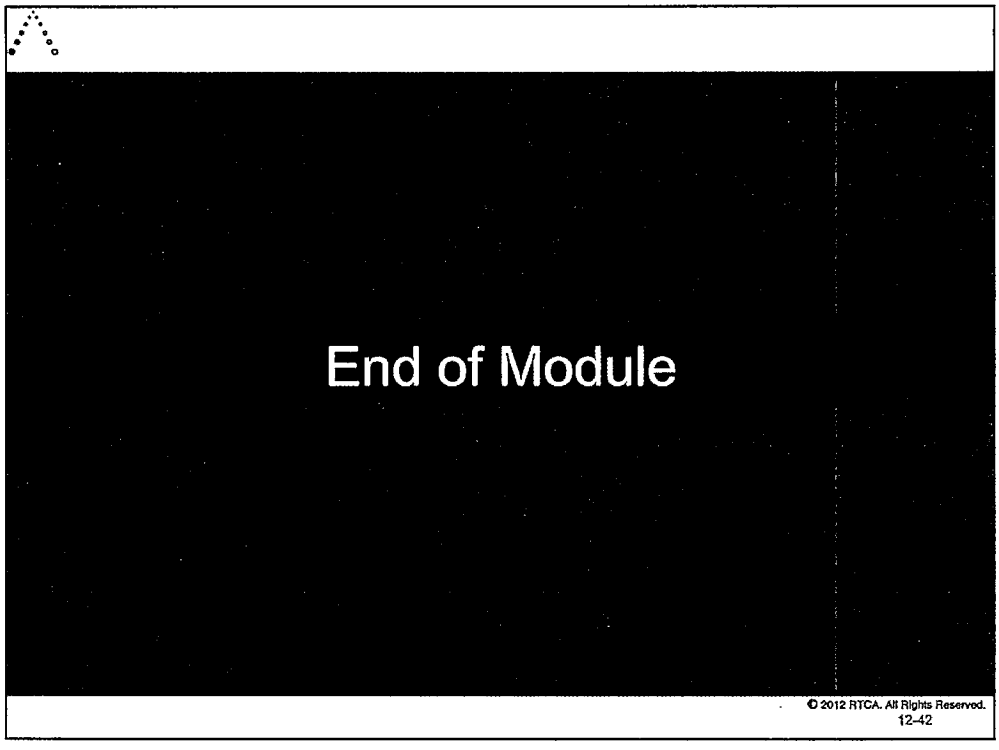
Each additional consideration can:

change objectives, activities and evidence,

add to them, and

delete objectives, activities and evidence.

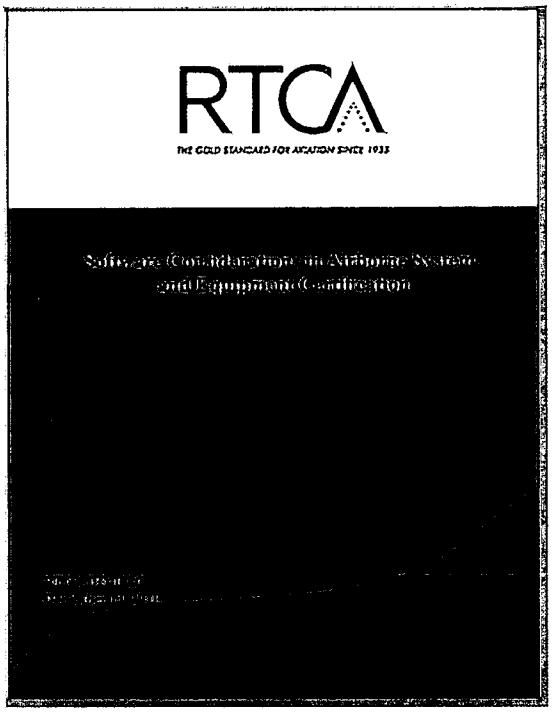
For Level D systems, service history is definitely possible



RTCA Software Developers' Course: *Software Considerations in Airborne Systems and Equipment Certification*

Module 13
Course Review and Summary

This courseware was developed by The MITRE Corporation for RTCA



RTCA, Inc.
1150 18th Street, NW
Suite 910
Washington, DC 20036-3816
USA

Module Objective

- At the end of this module, the participant will be able to:
  - understand the course materials through completing a review exercise
  - evaluate the course
  - receive the Course Certificate of Completion

Module Outline

- Course Review, Summary and Wrap Up
- Course Evaluations
- Certificates of Completion

Course Review Exercise Objective

- By reviewing all course materials, course participants will be able to describe the software considerations in airborne *systems* and equipment certification

13-5

Course Review Exercise Instructions

- Divide participants into up to 6 color-coded teams
- Each team is assigned a color-coded topic(s)
- Each team will choose its own software project and produce examples to support the questions assigned in the next slides
- Teams may use all available documents to develop their examples
- Out briefing method is team decision
- Exercise Duration: 45 minutes

Team Name	Assigned Topic(s)
Red Team	1. *System* Aspects Relating to Software Development
Orange Team	2. Software Life Cycle
	3. Software Planning Process
	4. Software Development Processes
Green Team	5. Software Verification Process
Blue Team	6. Software Configuration Management Process
	7. Software Quality Assurance Process
	8. Certification Liaison Process and Overview of Certification
Purple Team	9. Software Life Cycle Data
Gray Team	10. Additional Considerations

Red Team #1: Sample Topics

1. What is the scope of the term "*system*" as it is used in DO-178C?
2. What *system* data flows to the software life cycle processes?
3. What software data flows to the *system* life cycle processes?
4. What data flows between the software life cycle processes and the hardware life cycle processes?
5. Which Software Level is the least severe, in terms of failure condition, for an aircraft: A, B, C, D, or E?
6. Name 3 of the *system* architectural influences on software.

Orange Team #2: Sample Topics

7. Which software process "Defines and coordinates the activities of the software development and integral processes for a project"?
8. Identify the mechanism which allows the software process to obtain feedback from two directions: Received from other processes and Sent to other processes.
7. What software plans and standards have to be produced?
8. Distinguish the differences in content between the following software plans:

   Software Verification Plan

   Software Configuration Plan
9. If these plans are applied incorrectly, will they still produce DO-178C-compliant software? Why or why not?
10. Identify some possible transition criteria between the Software Design Process and the Software Coding Process.
11. Is a trace matrix required to achieve traceability activities?

Green Team #3: Sample Topics

13. What are some Software Testing Objectives for the Executable Object Code?
14. What trace data is required for software verification?
15. What data item(s) are all Test Cases created from?
    A. *system* requirement(s)
    B. source code
    C. software requirement(s)
    D. equivalence class(es)
    E. Data Flow(s)

Blue Team #4: Sample Topics

16. When should configuration management of requirements start?
17. Explain the chicken and egg problem:

    The software configuration index must include the software accomplishment summary.

    The software accomplishment summary must reference the software configuration index.

    How do you catalogue the catalogue?
18. The SQA process outputs in Table A-9 are ___?

    CC1

    CC2
19. SQA activities satisfy the SQA process by:

    _____

    _____

    _____
20. What key items does a Software Conformity Review determine?
21. Who establishes the Certification Basis for a *system* or piece of equipment?
22. What Software Life Cycle Data must be submitted to the Certification Authority for a certification?

Purple Team #5: Sample Topics

23. Which Data Item contains the results of source code reviews and analyses?

Gray Team #6: Sample Topics

24. What are "additional considerations"?
25. When would, and how should, sections in "additional considerations" be used?
26. How do "additional considerations" affect the objectives, activities, and considerations within DO-178C, sections 3 through 11?

Begin Exercise

Image Source: MS

13-13

Image Source: MS

TEAM OUT BRIEFS

13-14

Course Summary

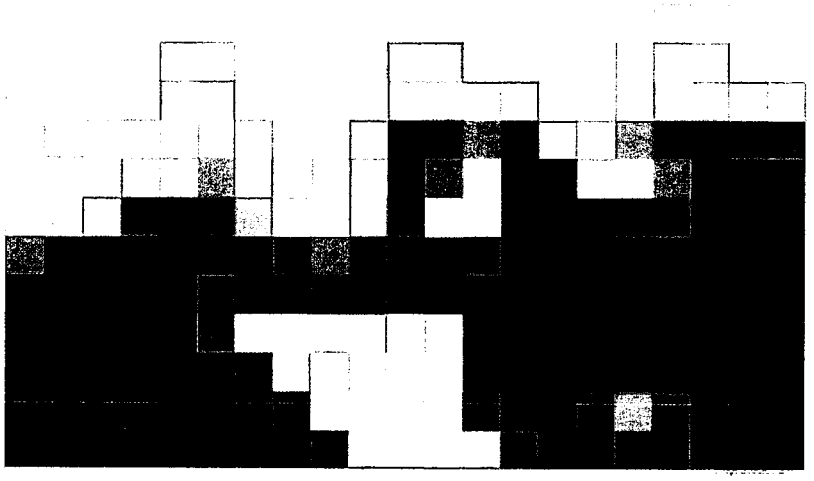
- You were provided with, and should now have a thorough understanding of, _____
  - ✓ An overview of RTCA DO-178C
  - ✓ A thorough understanding of the fundamental techniques for software assurance of airborne *systems* and equipment
  - ✓ The ability to understand and determine the applicability of DO-178C objectives based on software level
  - ✓ The ability to understand how to employ DO-178C to allow industry to comply with the document to assure the complete and correct implementation of the software requirements
  - ✓ An understanding of the objectives, activities, and life cycle data

COURSE EVALUATION

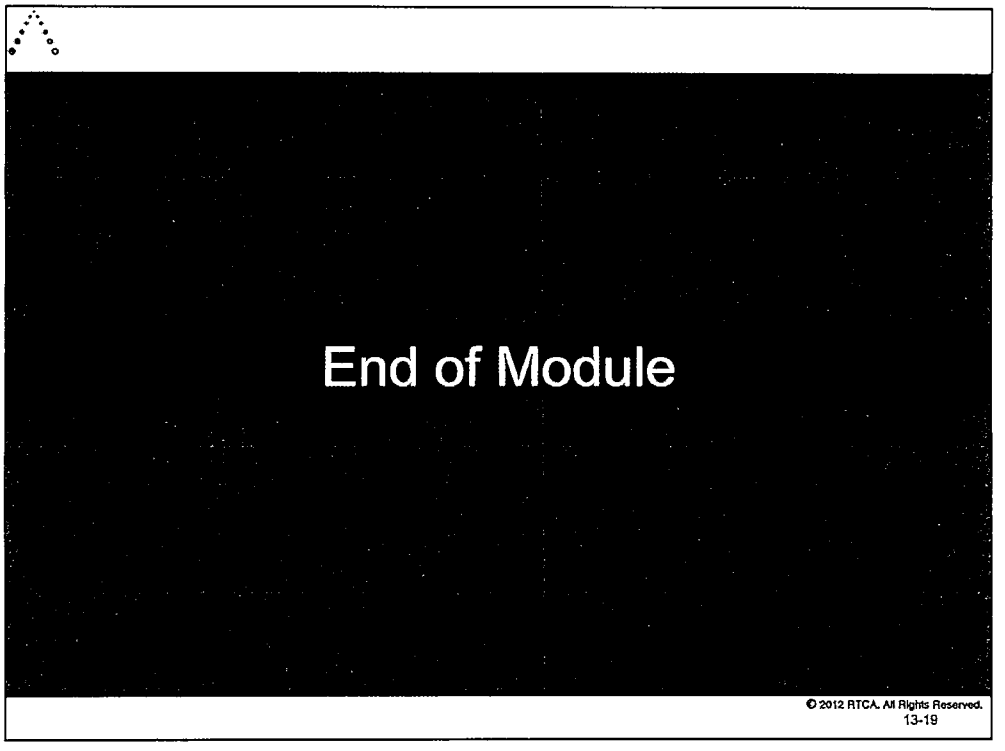
Overall Course Evaluation

- A course evaluation form is provided
- Please complete this form in order to provide MITRE Aviation Institute Instructors and RTCA feedback on:
  - Course activities
  - Instructors
  - Ways and means to improve the training

Graduation



13-18


End of Module
© 2012 RTCA. All Rights Reserved.
13-19