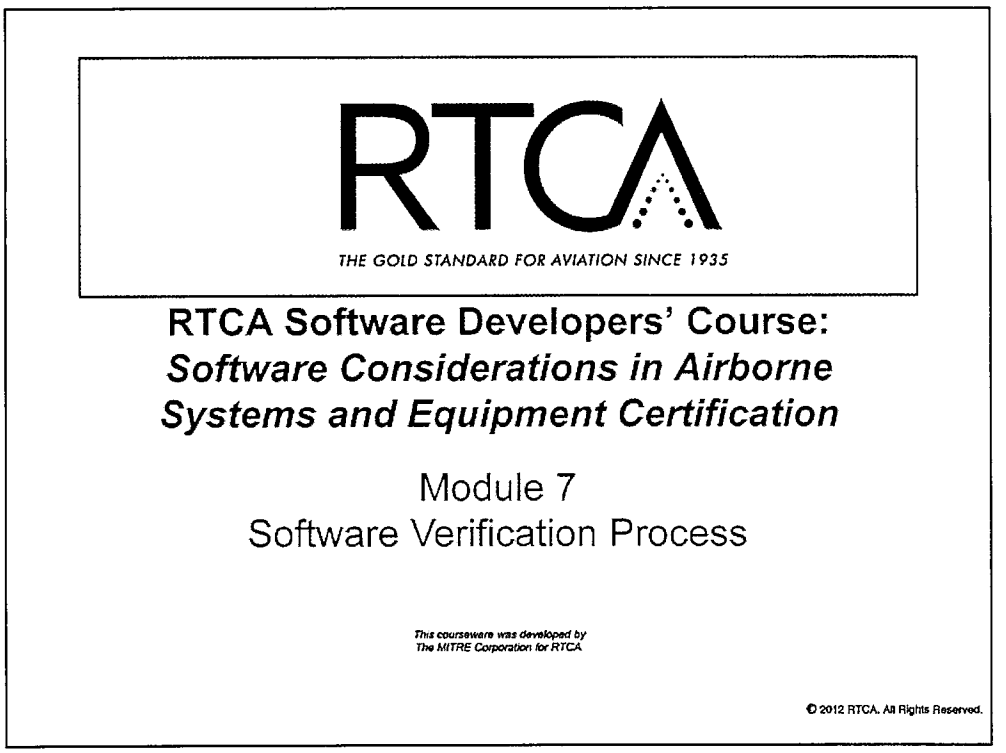


This is a two part, 5-hour module. In the first part, Review, Analysis and Software Testing are described. Also, the concepts of Test Cases, Test Procedures, and Test results are explained.

Verification Methods: More in-depth study of Review, Analysis and Testing methods.

The requirements-based test development process is described in detail. Test cases are created from software requirements, test procedures are created from test cases and test results from the execution of the code.

An in-depth discussion on Test Case selection for normal and robustness testing is provided.

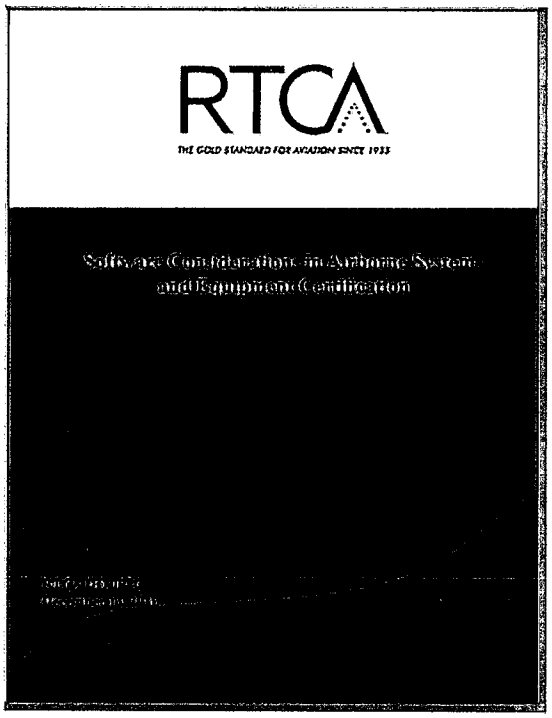
The second part of this module is an in-depth analysis of the verification of the testing results. This part is composed of the following content:

Coverage: This segment describes the process of structural coverage analysis from requirements based tests.

Review and analysis of test cases, procedures, and results

Trace Data

Throughout this module, participants will do exercises on reviews, creation of test cases, and structural coverage analysis. Exercises include using evaluation boards and a training-embedded application.



Section 6

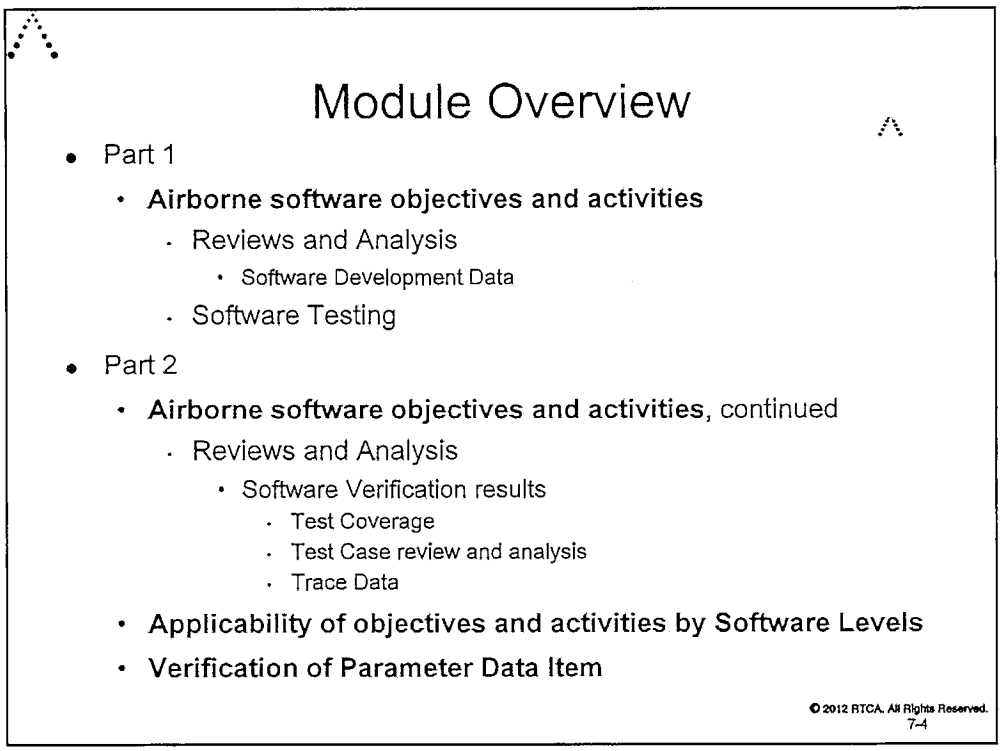
Module Objectives

At the end of this module the participant will understand, and be able to apply, through two exercises:

- Airborne software objectives and activities for:
  - Reviews and analysis (High-level Requirements, Low-level Requirements, and Source Code)
  - Testing
  - Test Coverage
  - Test Case Review and Analysis
- What trace data is required for Requirements, Source Code, Test Cases, Procedures, and Results.
- Applicability of objectives and activities by Software Level

This chart reviews the objectives of Module 7.

The goal is to implement the requirements.

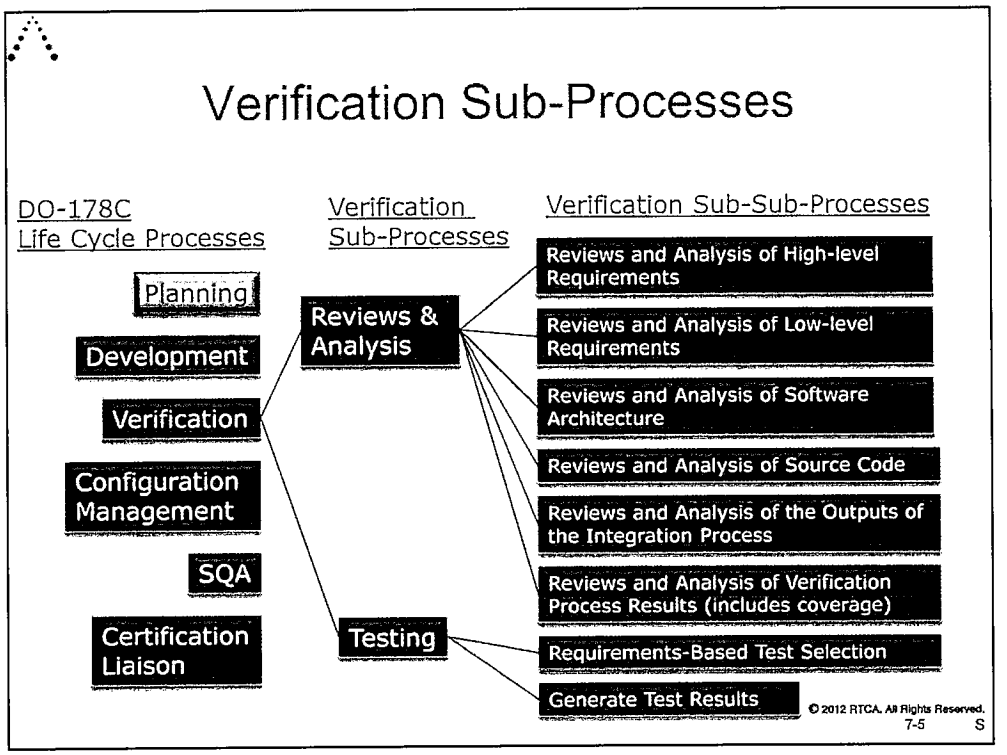


This chart shows the outline of Module 7. It is a two part module.

Testing means building test cases and executing the Executable Object Code to obtain actual results .

Test coverage – means

- do the test cases fully test the implementation of the requirements?
- Do the test cases cause all the code to be executed when the test cases are executed using the Executable Object Code?
- Do the test cases fully test all data transfers between code elements and test all required control of execution order.

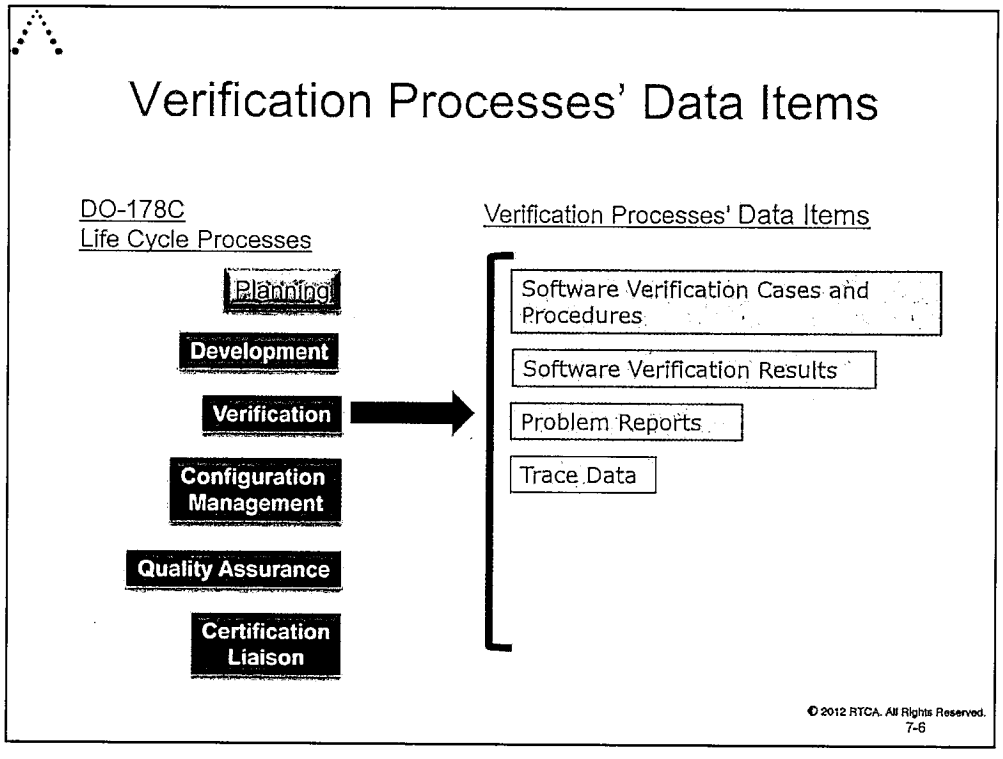


We will be covering the first five sub-sub-processes first.

Then we will cover testing.

Then, after testing, we will be covering the reviews and analysis of the verification process results.

The flow of this module is not in the order of the Sub-Sub-processes in the slide.

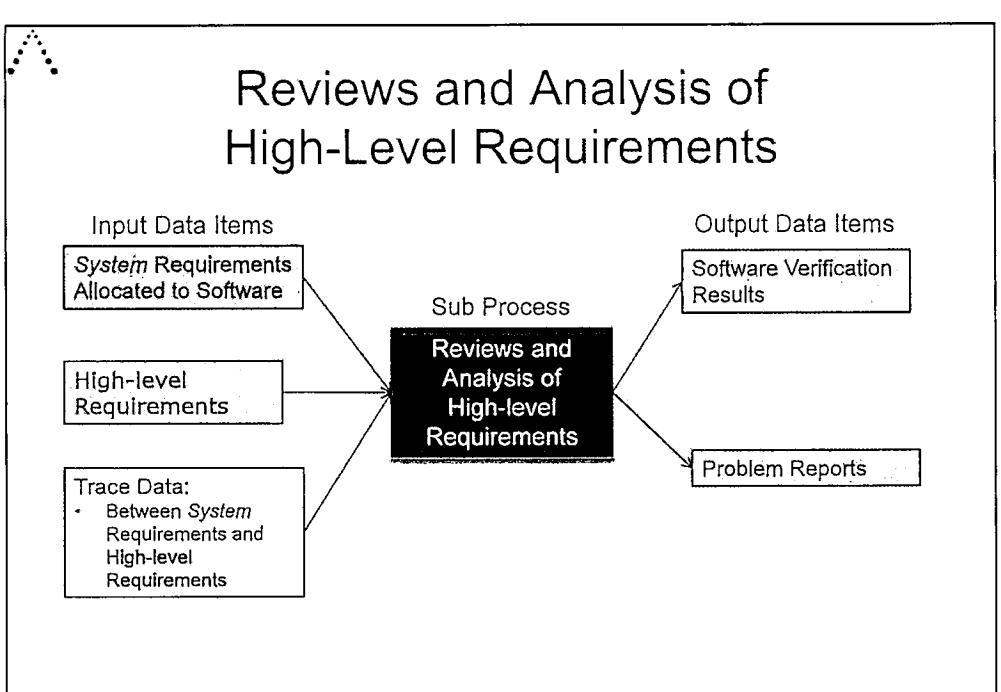


This chart shows all Software Life Cycle Data Items that are created during the over all verification process. Some of these items will be used by many of the sub-processes.

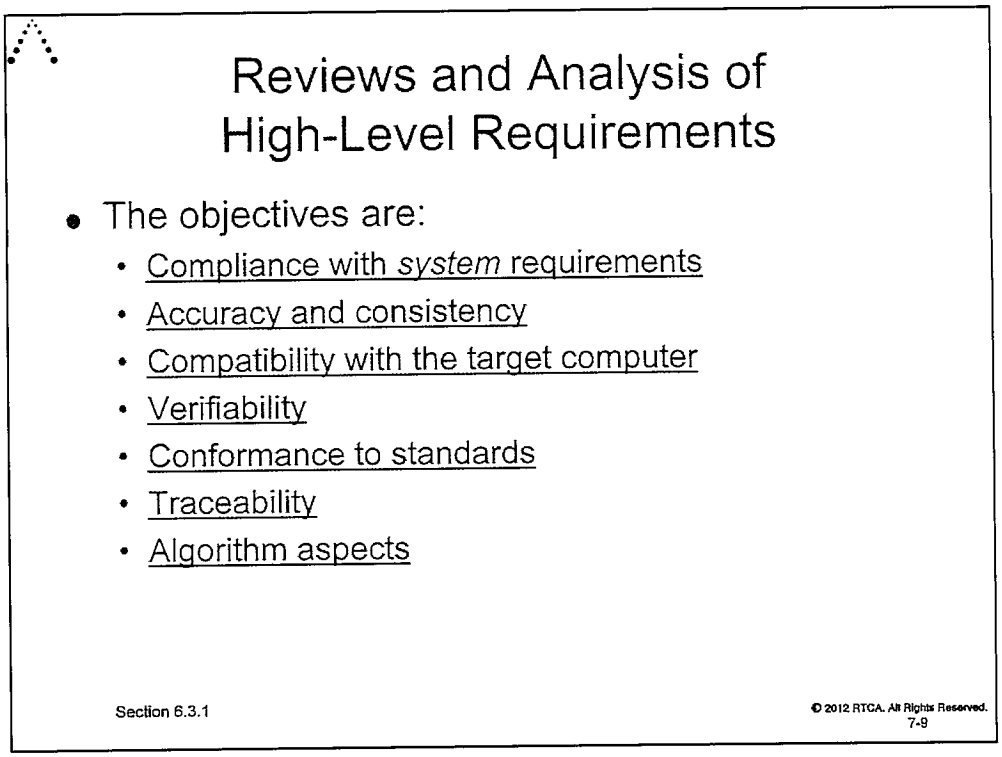
Section
6.3

Software Development Data

REVIEWS AND ANALYSIS



The review or analysis should be done following the planned process in the verification plan. Additionally, this process must consider the "development standard" for requirements.



These objectives imply existence of two software life cycle data item.

Compliance with *system* requirements: The objective is to ensure that the *system* functions to be performed by the software are defined, that the functional, performance, and safety-related requirements of the *system* are satisfied by the High-level requirements, and that derived requirements and the reason for their existence are correctly defined.

Accuracy and consistency: The objective is to ensure that each High-level requirement is accurate, unambiguous, and sufficiently detailed, and that the requirements do not conflict with each other.

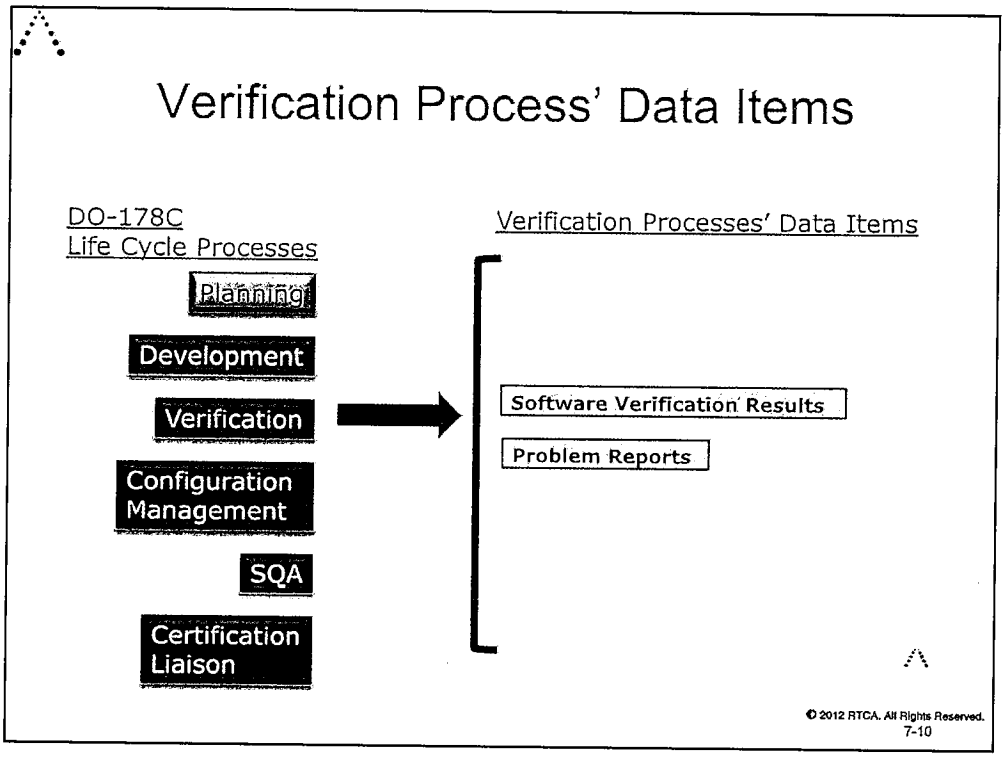
Compatibility with the target computer: The objective is to ensure that no conflicts exist between the High-level requirements and the hardware/software features of the target computer, especially *system* response times and input/output hardware.

Verifiability: The objective is to ensure that each High-level requirement can be verified.

Conformance to standards: The objective is to ensure that the Software Requirements Standards were followed during the software requirements process and that deviations from the standards are justified.

Traceability: The objective is to ensure that the functional, performance, and safety-related requirements of the *system* that are allocated to software were developed into the High-level requirements.

Algorithm aspects: The objective is to ensure the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities.

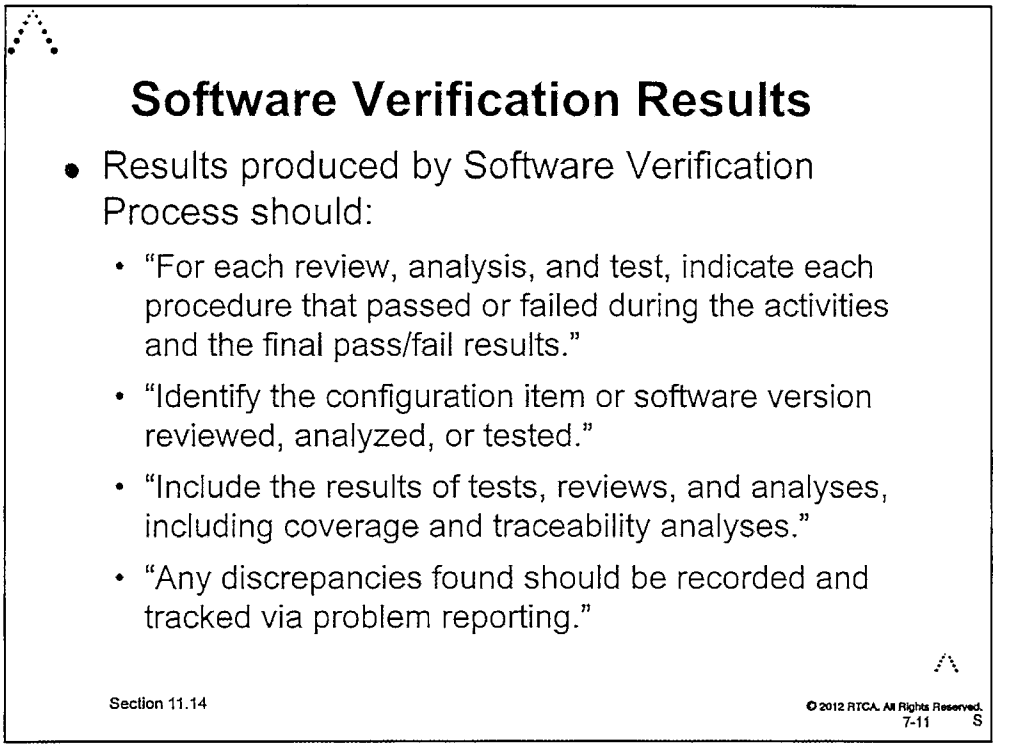


See Second Screen slide titled:

- Software Verification Results

And slide near end of this module titled:

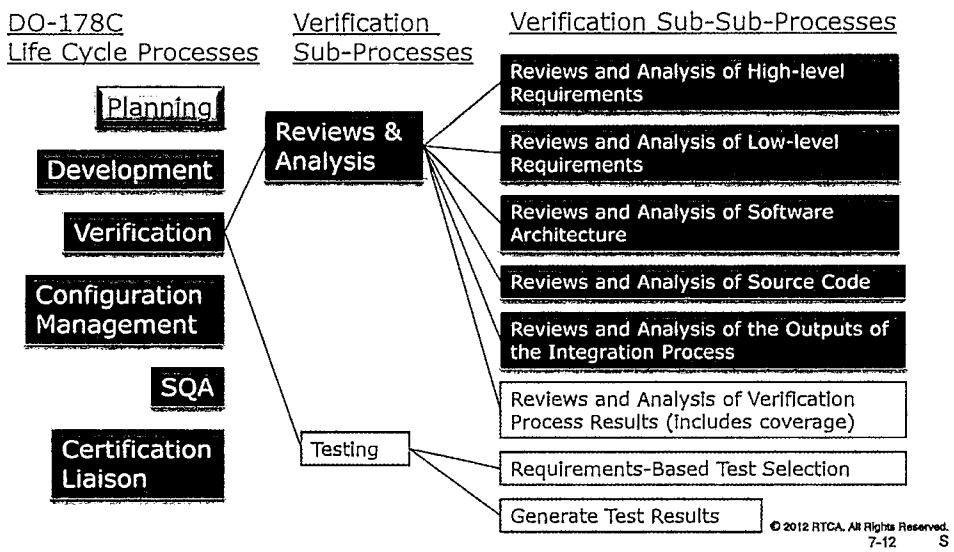
- Problem Reports

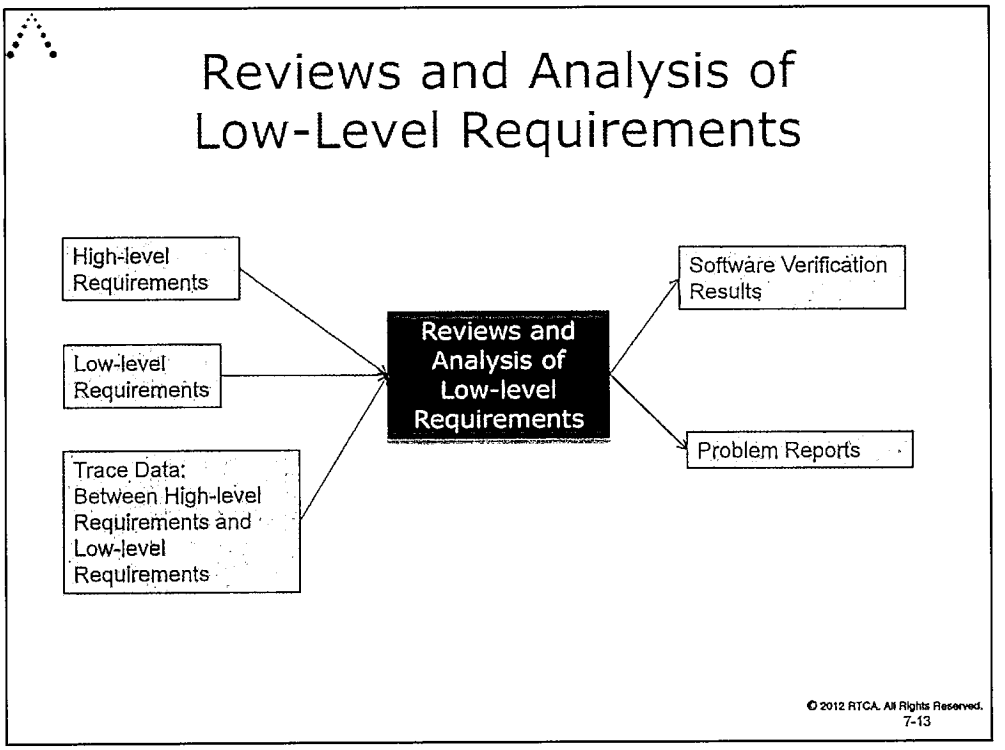


This is a catch-all data item for all verification results. It may encompass many different physical documents and formats. Some, even most, formats may be electronic.

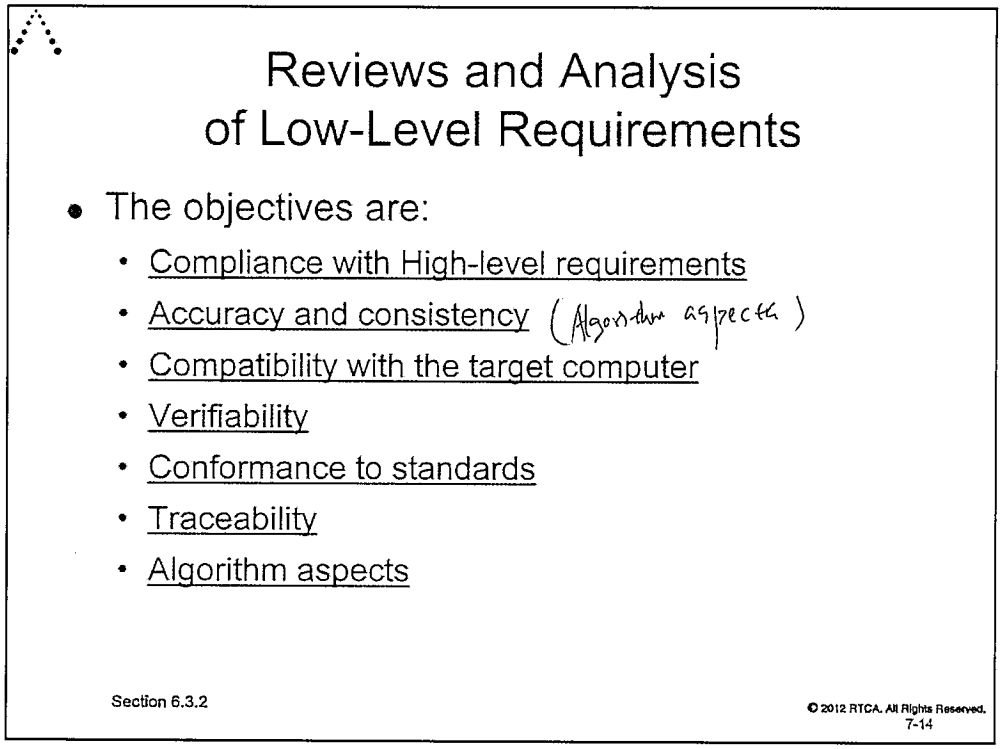
A good idea is to link a failure of a verification activity to the problem report that documents that failure. This provides additional information when problems are being resolved.

Verification Sub-Processes





The review and/or analysis should be done following the documented process in the verification plan. Additionally, this process must consider the "development standard" for requirements. When High-level requirements are used, they include High-level derived requirements. The same convention is used for Low-level requirements,.



These objectives imply existence of two software life cycle data item

Compliance with High-level requirements: The objective is to ensure that the Low-level requirements satisfy the High-level requirements and that derived requirements and the design basis for their existence are correctly defined.

Accuracy and consistency: The objective is to ensure that each Low-level requirement is accurate and unambiguous, and that the Low-level requirements do not conflict with each other.

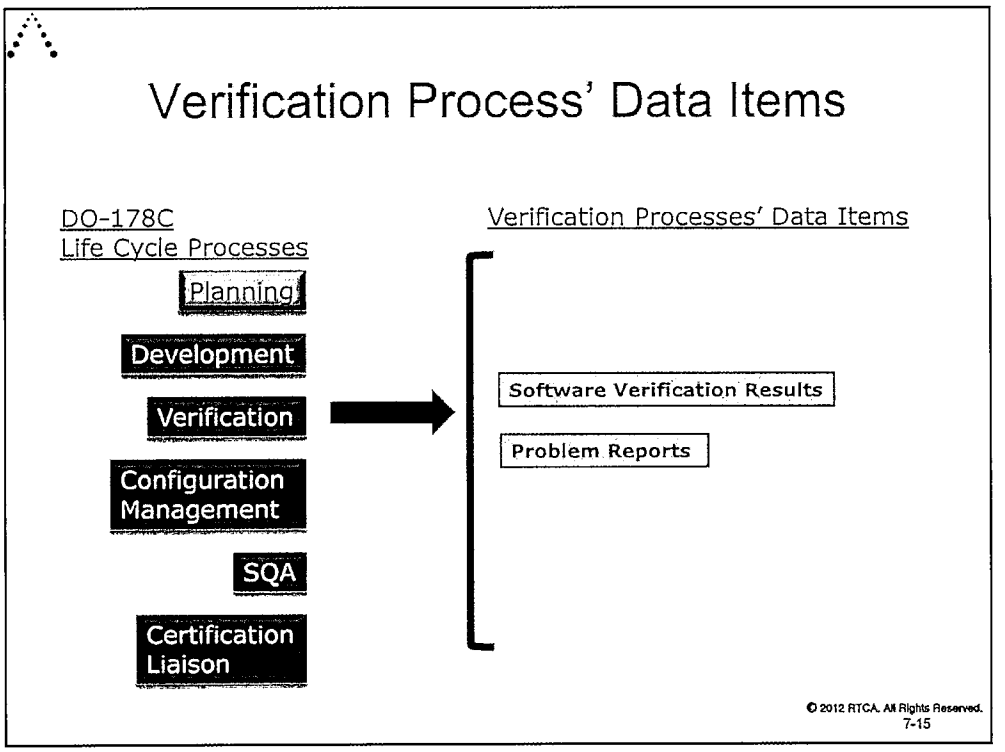
Compatibility with the target computer: The objective is to ensure that no conflicts exist between the Low-level requirements and the hardware/software features of the target computer, especially the use of resources such as bus loading, *system* response times, and input/output hardware.

Verifiability: The objective is to ensure that each Low-level requirement can be verified.

Conformance to standards: The objective is to ensure that the Software Design Standards were followed during the software design process and that deviations from the standards are justified.

Traceability: The objective is to ensure that the High-level requirements and derived requirements were developed into the Low-level requirements.

Algorithm aspects: The objective is to ensure the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities.

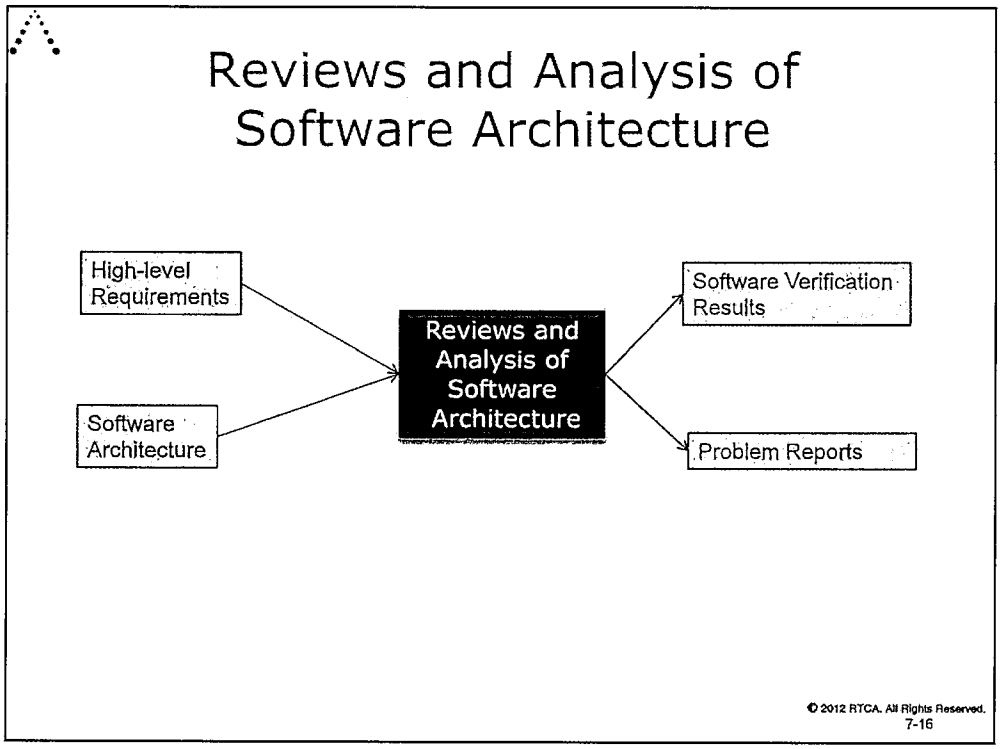


See Second Screen slide titled:

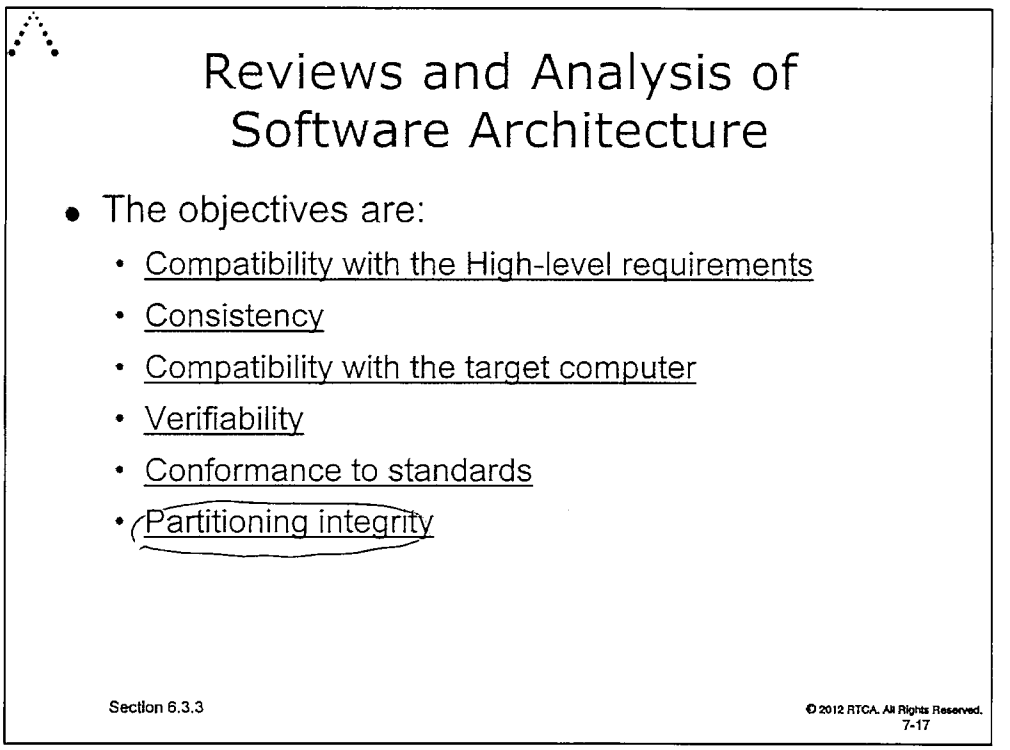
- Software Verification Results

And slide near end of this module titled:

- Problem Reports



The review or analysis should be done following the documented process in the verification plan. Additionally, this process must consider the "development standard" for design.



These objectives imply existence of two software life cycle data item

Compatibility with the High-level requirements: The objective is to ensure that the software architecture does not conflict with the High-level requirements, especially functions that ensure *system* integrity, for example, partitioning schemes.

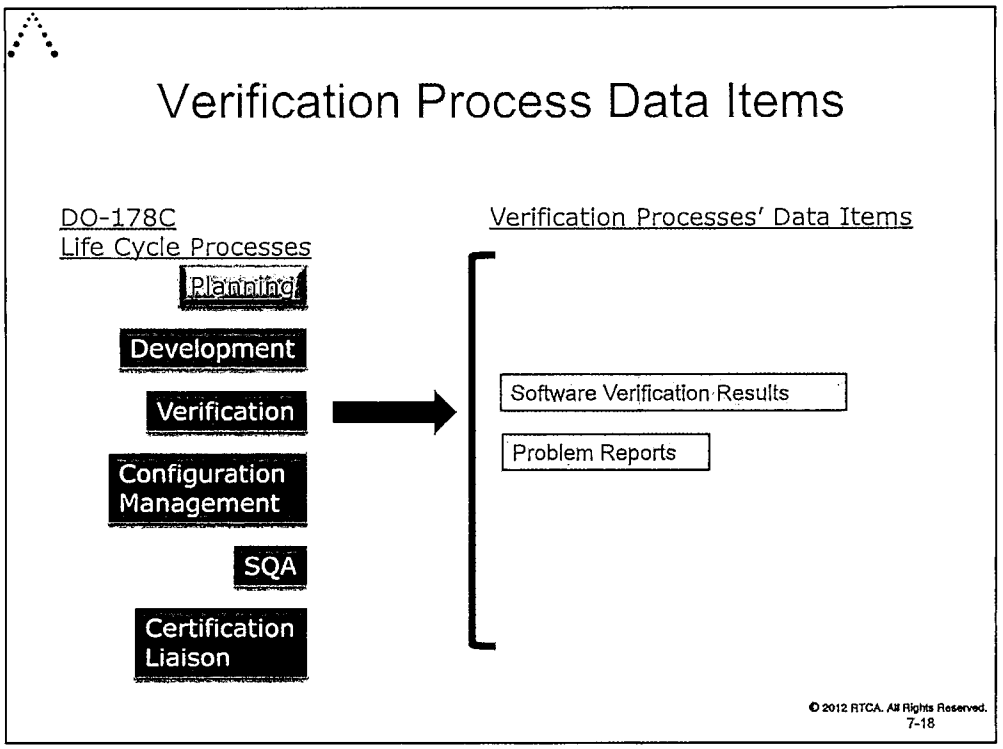
Consistency: The objective is to ensure that a correct relationship exists between the components of the software architecture. This relationship exists via data flow and control flow. If the interface is to a component of a lower software level, it should also be confirmed that the higher software level component has appropriate protection mechanisms in place to protect itself from potential erroneous inputs from the lower software level component.

Compatibility with the target computer: The objective is to ensure that no conflicts exist, especially initialization, asynchronous operation, synchronization, and interrupts, between the software architecture and the hardware/software features of the target computer.

Verifiability: The objective is to ensure that the software architecture can be verified, for example, there are no unbounded recursive algorithms.

Conformance to standards: The objective is to ensure that the Software Design Standards were followed during the software design process and that deviations to the standards are justified, for example, deviations to complexity restriction and design construct rules.

Partitioning integrity: The objective is to ensure that partitioning breaches are prevented.



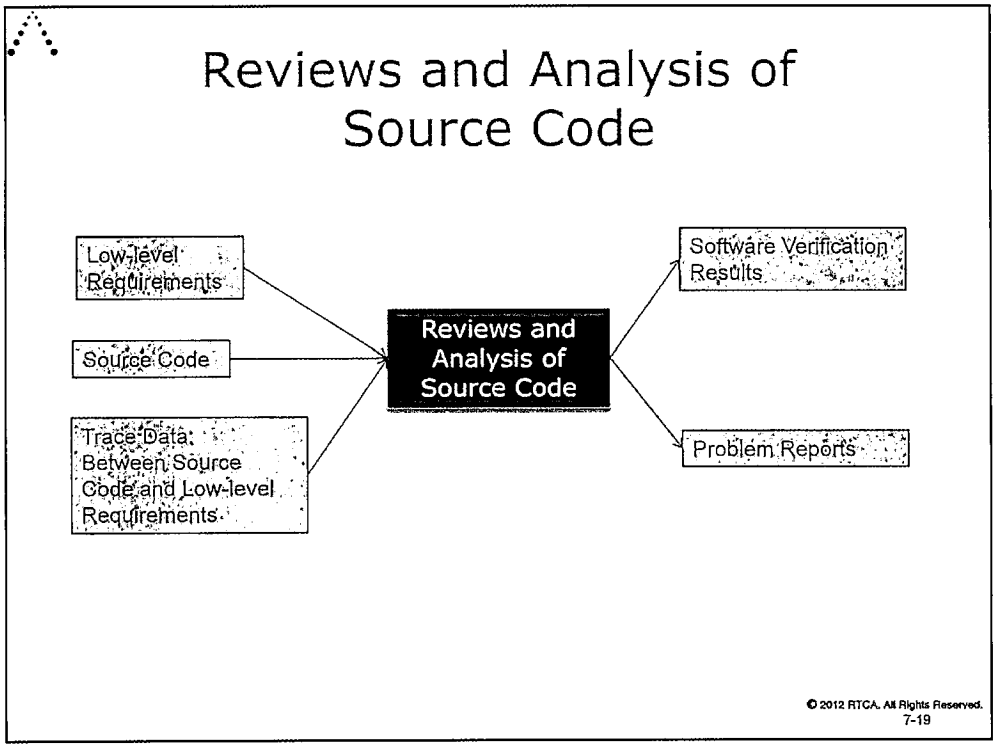
See Second Screen slide titled:

- Software Verification Results

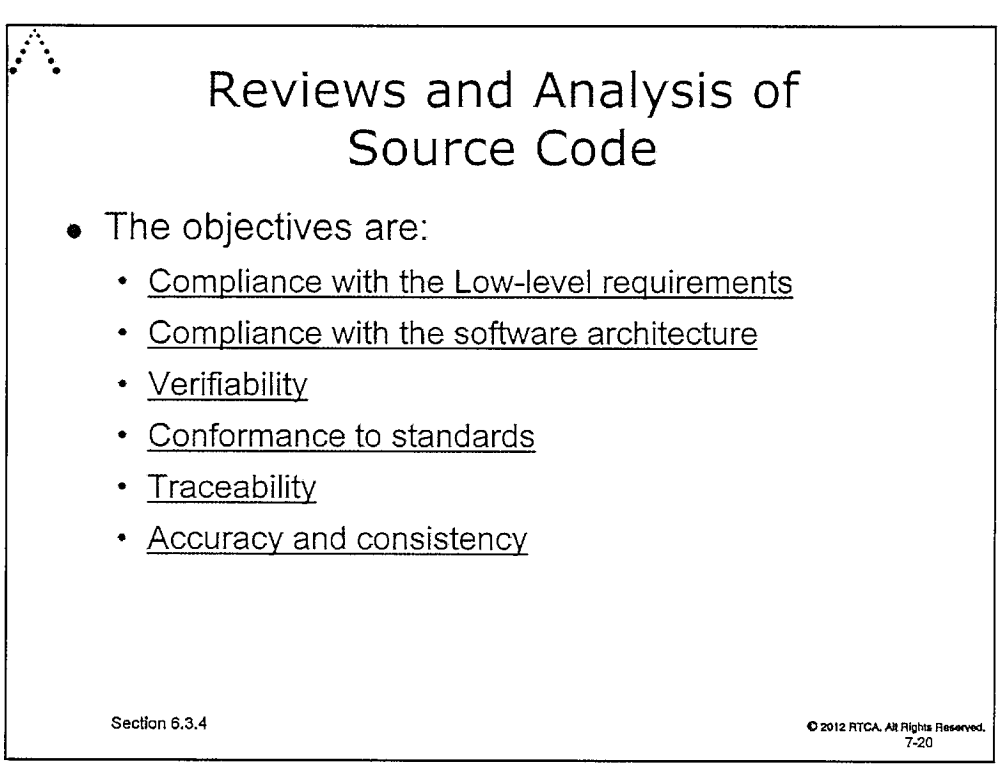
Problem reports data Item is covered in a later slide

See slide near end of this module titled:

- Problem Reports



The review or analysis should be done following the documented process in the verification plan. Additionally, this process must consider the "development standard" for coding.



These objectives imply existence of two software life cycle data item

Compliance with the Low-level requirements: The objective is to ensure that the Source Code is accurate and complete with respect to the Low-level requirements and that no Source Code implements an undocumented function.

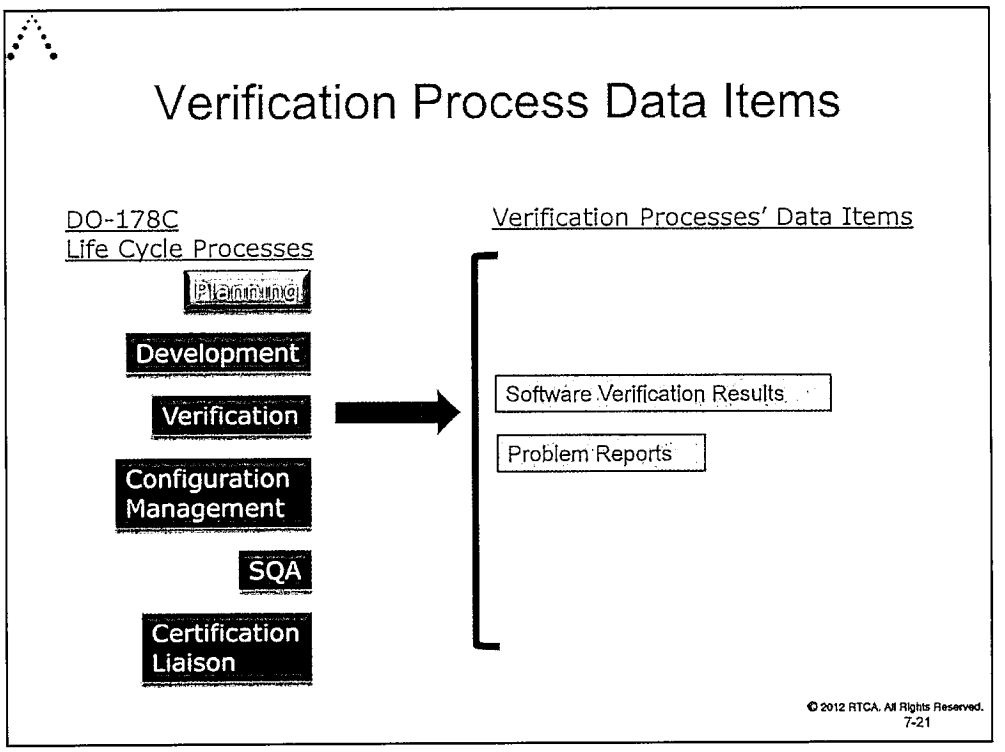
Compliance with the software architecture: The objective is to ensure that the Source Code matches the data flow and control flow defined in the software architecture.

Verifiability: The objective is to ensure the Source Code does not contain statements and structures that cannot be verified and that the code does not have to be altered to test it.

Conformance to standards: The objective is to ensure that the Software Code Standards were followed during the development of the code, for example, complexity restrictions and code constraints. Complexity includes the degree of coupling between software components, the nesting levels for control structures, and the complexity of logical or numeric expressions. This analysis also ensures that deviations to the standards are justified.

Traceability: The objective is to ensure that the Low-level requirements were developed into Source Code.

Accuracy and consistency: The objective is to determine the correctness and consistency of the Source Code, including stack usage, memory usage, fixed point arithmetic overflow and resolution, floating-point arithmetic, resource contention and limitations, worst-case execution timing, exception handling, use of uninitialized variables, cache management, unused variables, and data corruption due to task or interrupt conflicts. The compiler (including its options), the linker (including its options), and some hardware features may have an impact on the worst-case execution timing and this impact should be assessed.



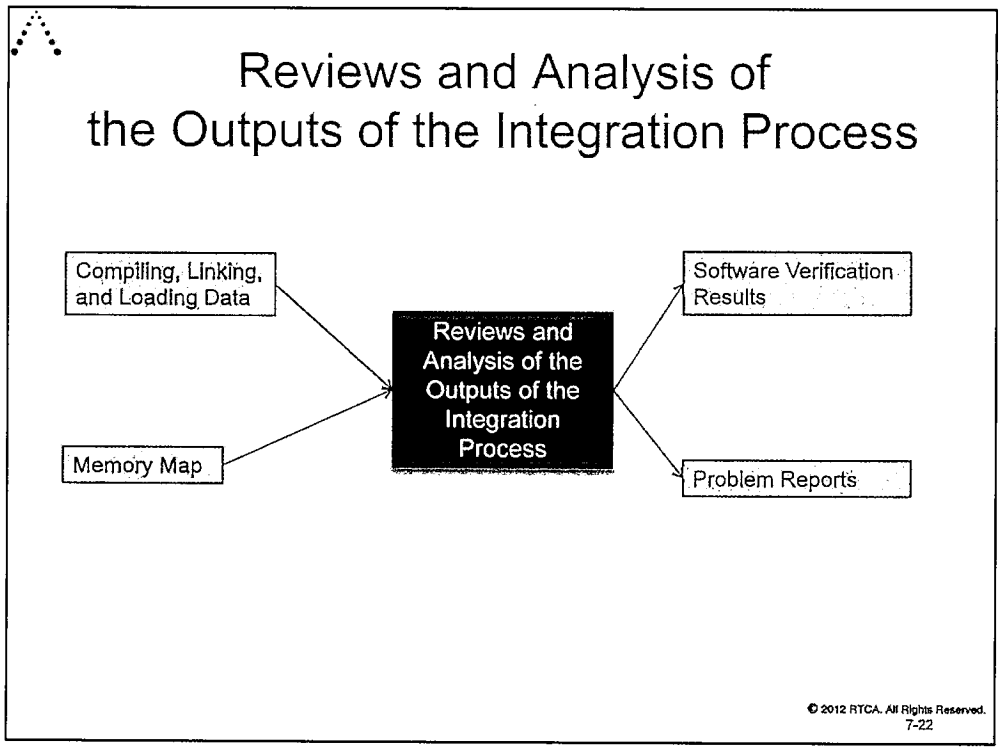
See Second Screen slide titled:

- Software Verification Results

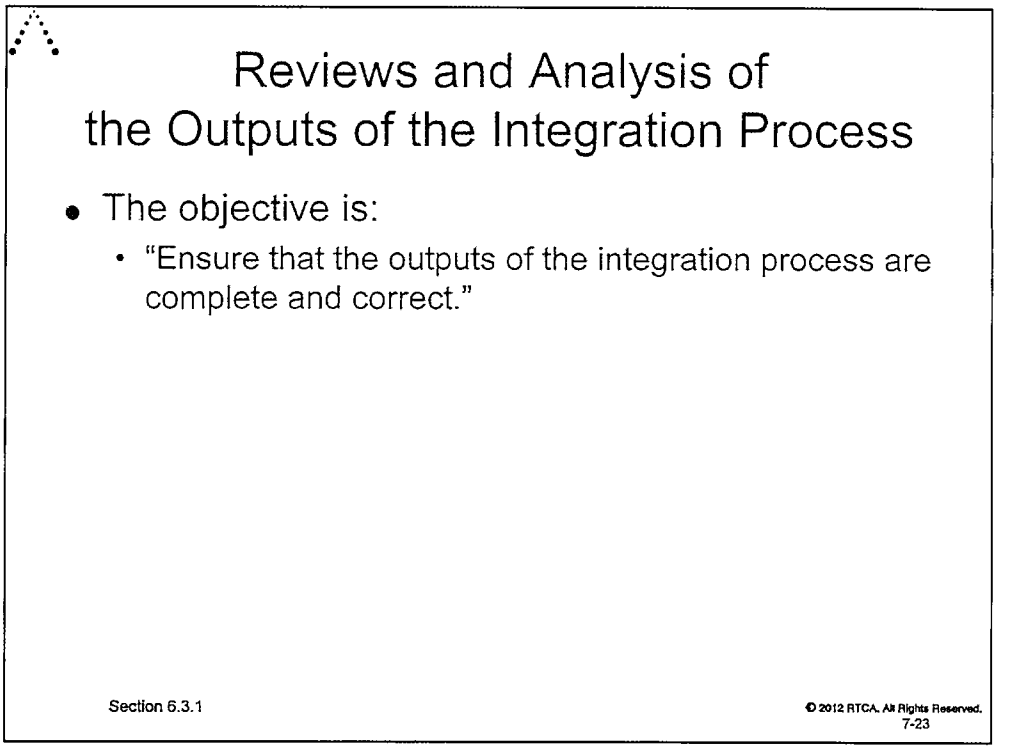
Problem reports data Item is covered in a later slide

See slide near end of this module titled:

- Problem Reports



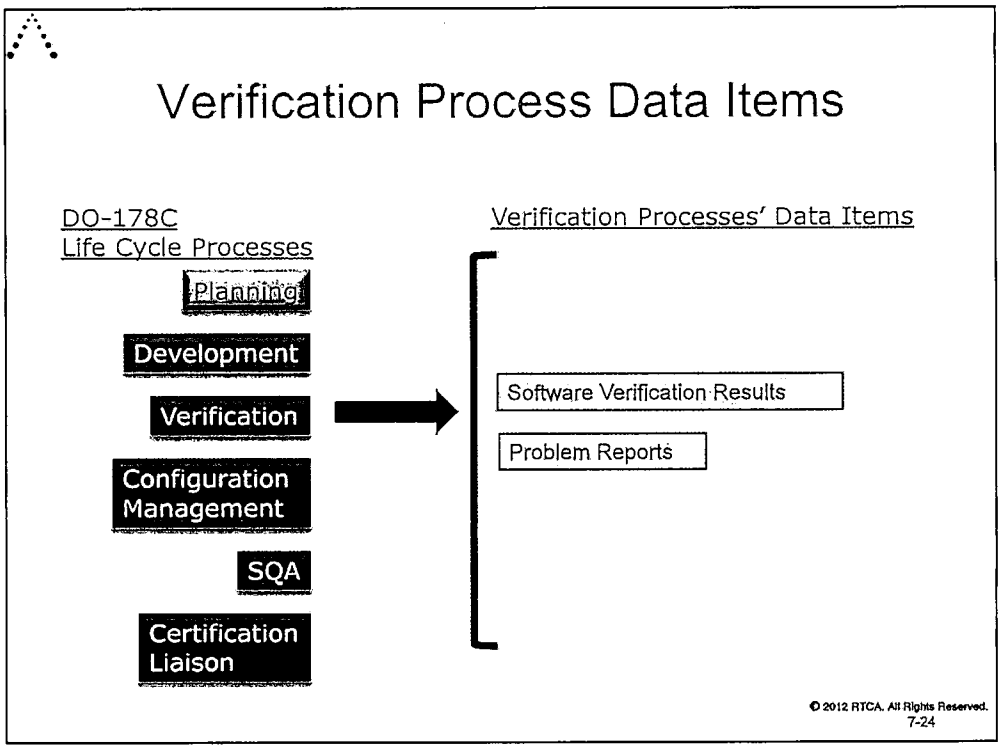
The review or analysis should be done following the documented process in the verification plan.



This activity includes conducting a detailed examination of the compiling, linking and loading data, and memory map.

Typical examples of potential errors include:

- Compiler warnings.
- Incorrect hardware addresses.
- Memory overlaps.
- Missing software components



See Second Screen slide titled:

- Software Verification Results

Problem reports data Item is covered in a later slide

See slide near end of this module titled:

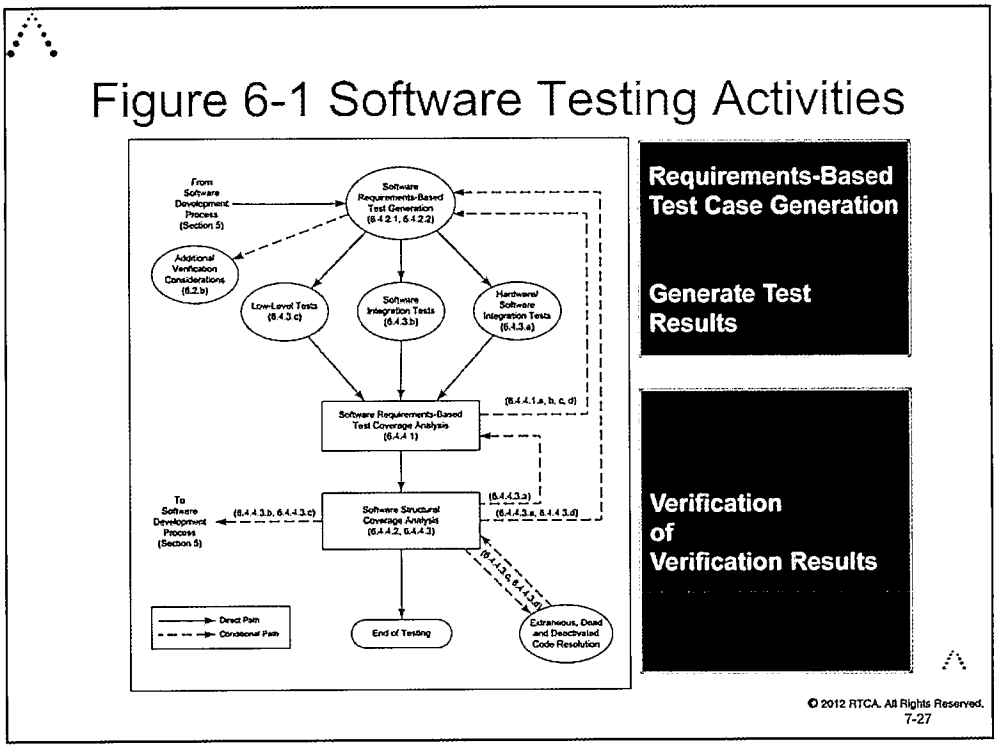
- Problem Reports

Exercise #4: Source Code Review

- Do a review of the Instructor-supplied Source Code using the checklist you created in Module 5 – "Code Review Checklist"
- Write down any problems you find because we will use them later to create a Problem Report
- Obtain Instructor-supplied Source Code for Table Look Up of Altitude Using Pressure

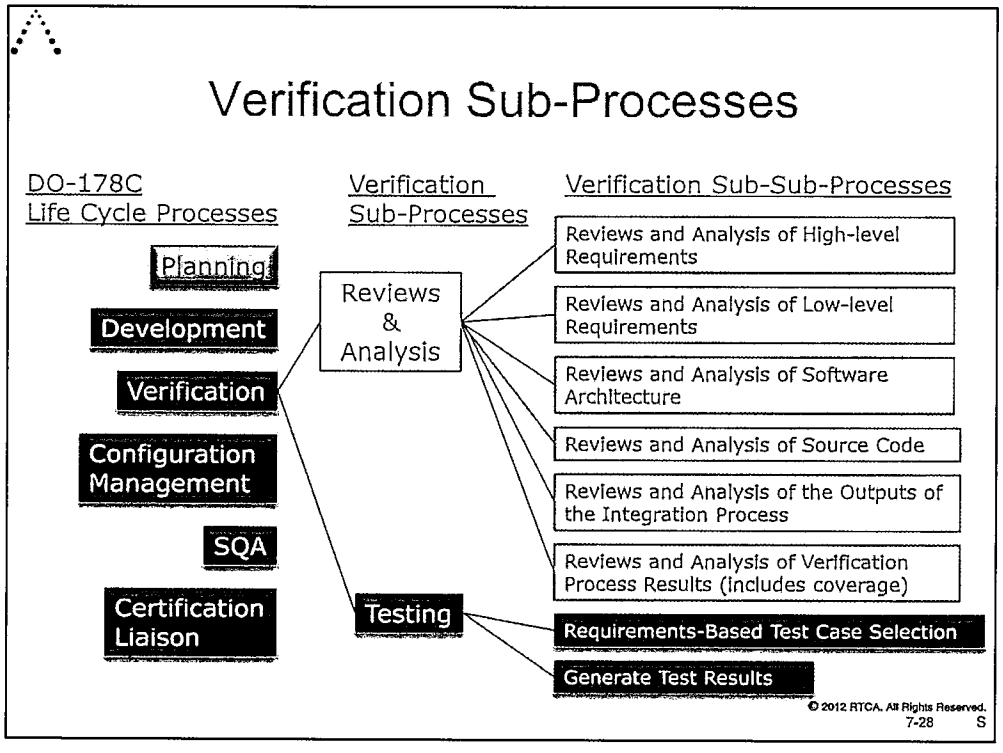
Section
6.4

SOFTWARE TESTING



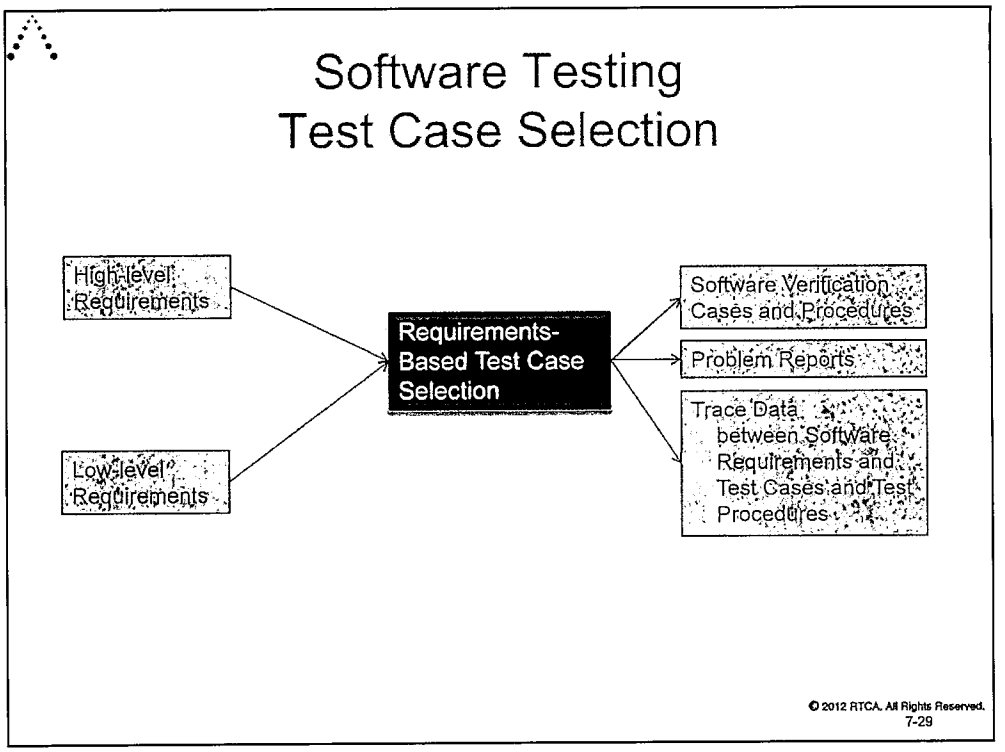
This figure is useful to help guide the testing work flow. Work flow may generally go from top to bottom but needs to be specified in the Verification Plan for the specific project.

The top half of the figure illustrates test case generation and testing. The bottom half is Verification of the Verification results. There are feedback loops from the bottom half to the top half which are used if problems are found.



The next set of slides cover the testing sub-process and its sub-sub-processes:

- Requirements-Based test selection (Our focus now)
- Generation of test results



Test Cases and procedures should be created by following the documented process in the verification plan.

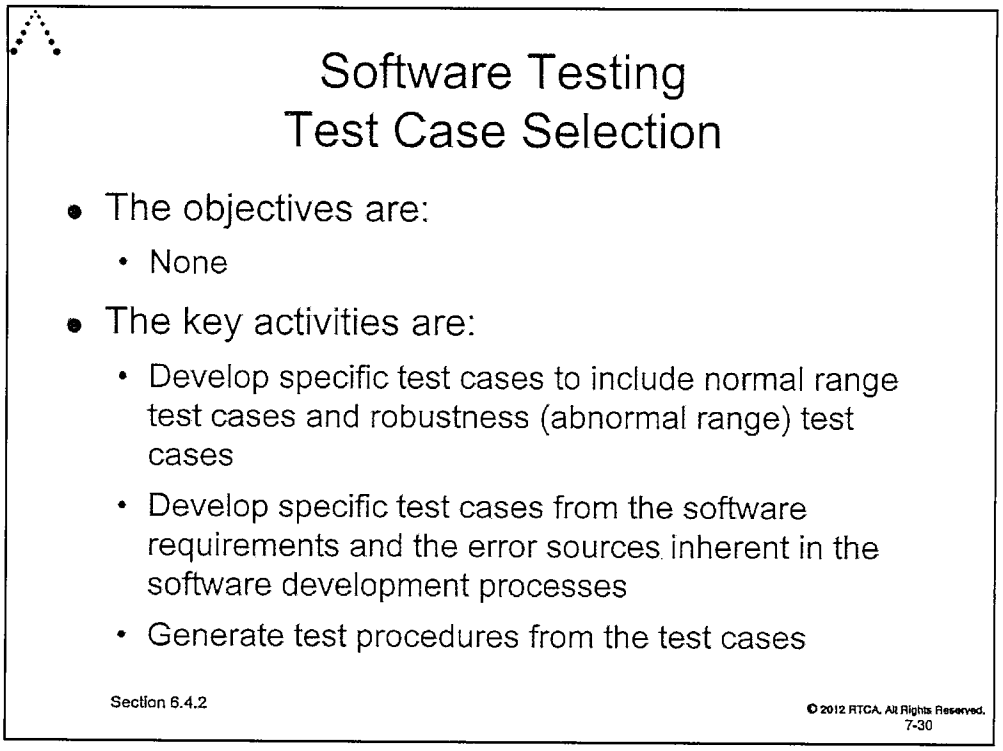
Note:

High-level Requirements includes Derived High-level Requirements

Low-level Requirements includes Derived Low-level Requirements

Test Procedures are the instructions for executing the test case on the hardware, emulator or simulator.

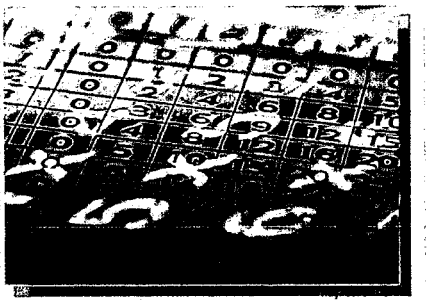
Test results are documentation of what happened when you execute the Test Procedures.



- There is no objective (as in the development process) that states that test cases and test procedures have to exist.
    - Since they will be reviewed, they **will have to exist** and be configuration management controlled.
    - The reason for the Testing Process is not to create test cases and procedures but to show that the implementation is correct through execution of the Executable Object Code in the target hardware.
    - Test cases and procedures are by-products of showing correct implementation. However they must be the correct by-products which is why there is a verification process of the verification results.
- Trace data needs to be created between requirements and test cases and between test cases and test procedures. It is not specified when this needs to be done. Avionics development experience supports doing it as the test cases and procedures are developed.

Software Testing
Test Case Selection

- Requirements-Based
- Normal Range Test Cases
  - Equivalence Classes and Boundary Values
  - Time
  - State transitions
  - Logical equations



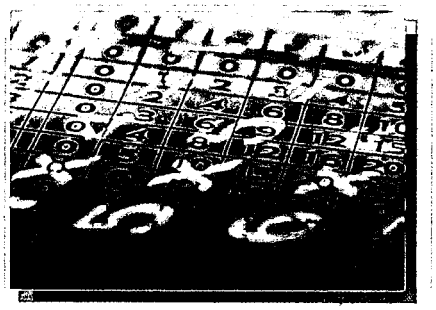
7-31

"The specific test cases should be developed from the software requirements and the error sources inherent in the software development processes." (section 6.4.2 item 2)

- Specific test cases should be developed to include normal range test cases. (section 6.4.2 item 1)
  - Normal range (see slide: Normal and Robust Test Cases)
- Exercise all possible and valid state transitions
- All possible combinations of the logic

Software Testing
Test Case Selection

- Robustness Test Cases
  - Equivalence Class
  - *System* initialization
  - Incoming data
  - Loops
  - Time
  - State Transitions



Robustness test cases are requirements-based.

7-32

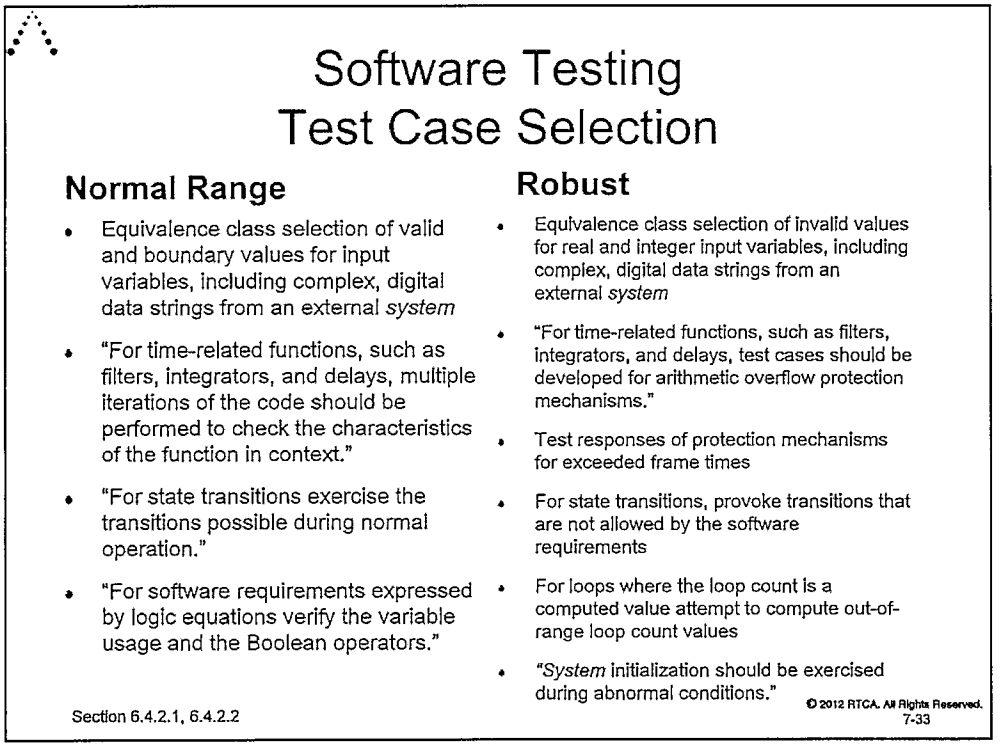
- Specific test cases should be developed to include robustness (abnormal range) test cases. (section 6.4.2 item 1)
  - Robust (see slide: Normal and Robust Test Cases)

"Note: Robustness test cases are requirements-based. The robustness testing criteria cannot be fully satisfied if the software requirements do not specify the correct software response to abnormal conditions and inputs. The test cases may reveal inadequacies in the software requirements, in which case the software requirements should be modified. Conversely, if a complete set of requirements exists that covers all abnormal conditions and inputs, the robustness test cases will follow from those software requirements." (section 6.4.2)

- This was added from DO-178B.
- Robustness test cases generally involve values that are out of range or abnormal in some way. How software reacts to them needs to be specified in requirements.
- While in may be argued that many "error sources inherent in the software development processes" are programmer decisions at the coding level, it is not a good practice to let a programmer decide what to do with an out-of-range value. Occasionally, errors can be introduced into the *system* at the coding level.

If you give bad data, you don't want it to die or do something unexpected.

Find the one input that breaks the system and makes it happen.



Example: Square Root function's input

- Normal range would be zero and any positive number(s) including the maximum positive number
- Robust would include any negative number(s) including the maximum and minimum negative numbers

Expected outputs depend on the input and output variable type.

What might be the equivalent classes (valid and invalid) for the Low-level requirements of our exercise "Create Low-Level requirements" in Module 6.

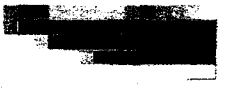
- Minimum Pressure
- Maximum Pressure
- Pressure at 0 altitude
- Pressure between 2 Altitudes
- Pressure less than Minimum
- Pressure greater than Minimum

There are two types of State Diagrams – Table and Connected Networked. Tables don't give completeness criteria. Test all possible flows in the network.

Specify all True/False, less than, greater than values in a comparison.

Software Testing Methods

- Hardware/Software Integration Testing
  - Best testing environment
  - Must run some of these tests
- Software Integration Testing
- Low-level Testing



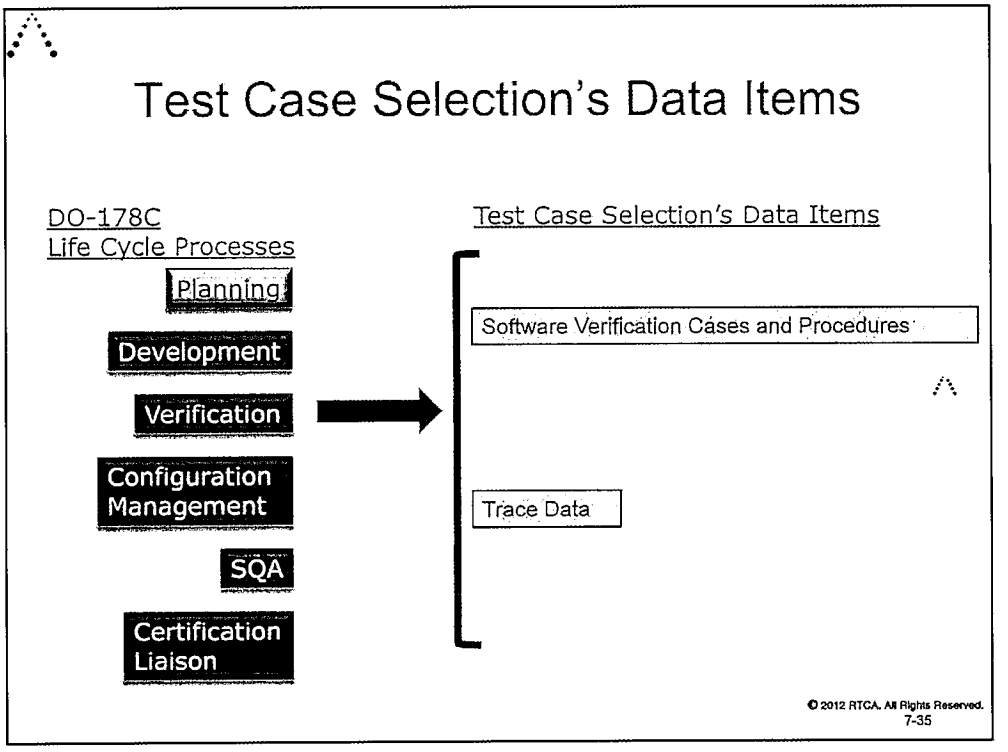
7-34

DO-178C lists three methods of testing.

DO-178C states "A preferred test environment includes the software loaded into the target computer and tested in an environment that closely resembles the behavior of the target computer environment". This is the Hardware/Software Integration Testing.

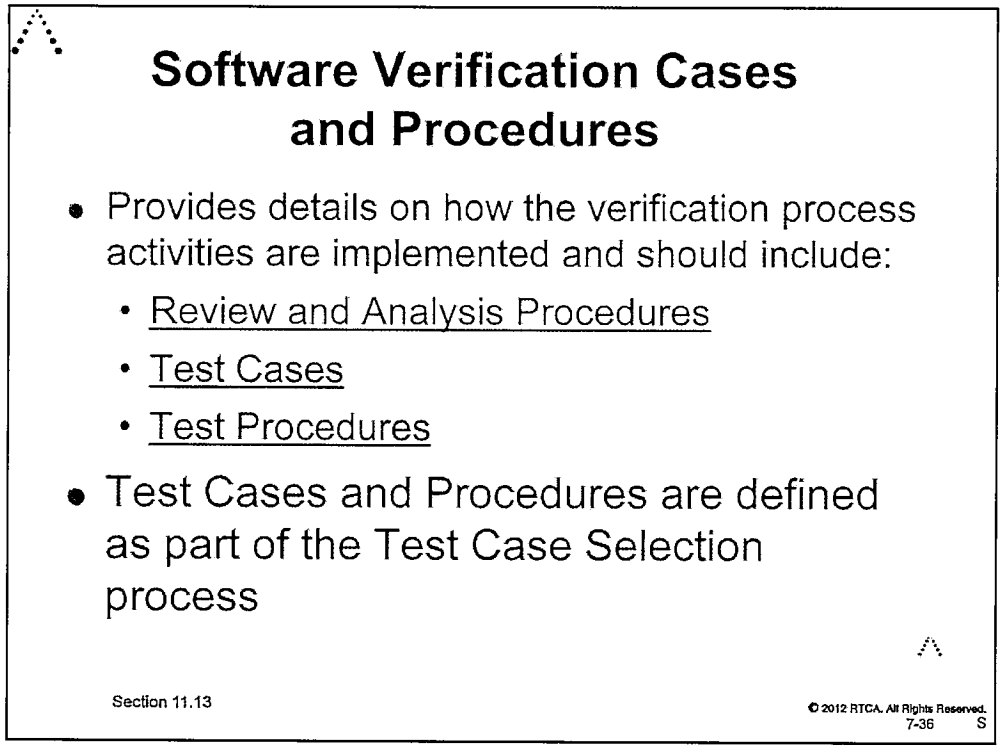
Generally for the other two methods a simulator might be used. The processor (target hardware) in a test setup might be used with only part of the final software loaded.

A simulator has to be qualified as a good simulator. Tool qualifications will be discussed tomorrow.



See Second Screen Slides titled:

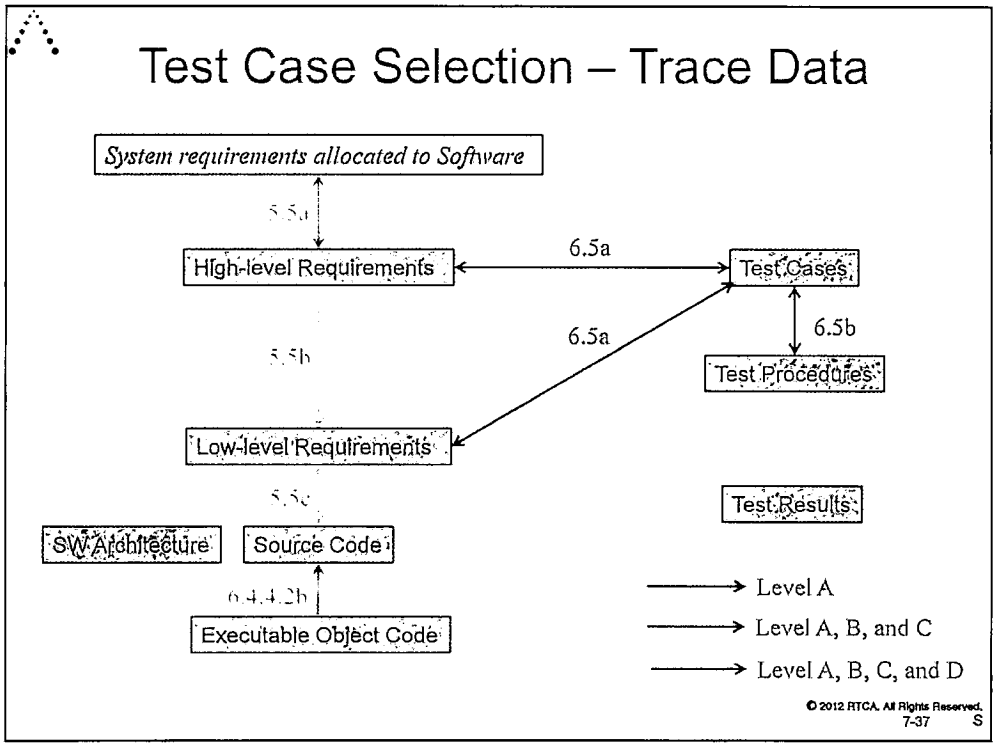
- "Software Verification Cases and Procedures"
- "Trace Data"



Test case – A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

Test cases need to specify the input and expected results for what is being tested. They also need to trace to the requirements upon which they are based.

Test procedures are the steps needed to execute the test case on the Executable Object Code.



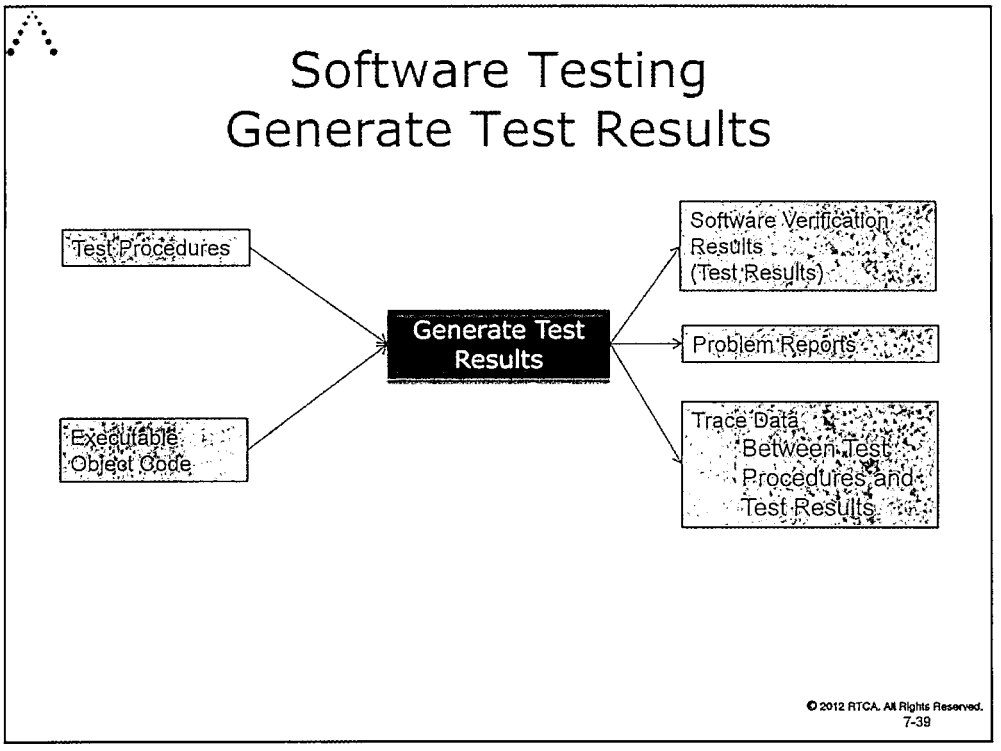
The following trace data is best created during test case generation.

Again note that identification names may be used as trace data if naming conventions allow.

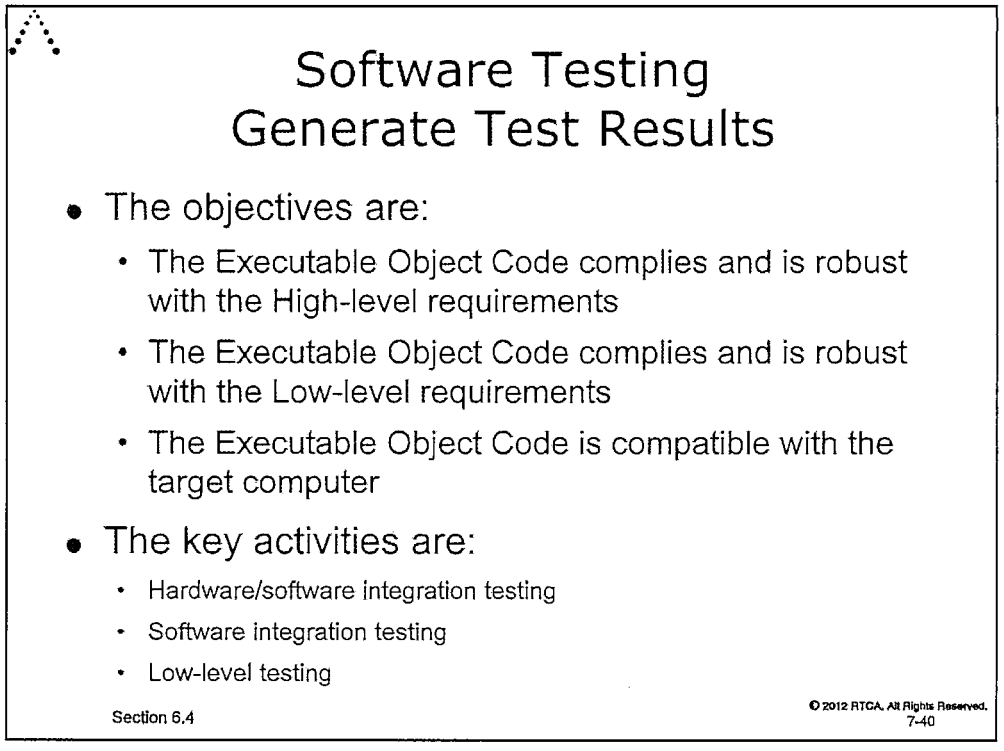
JOHN: Look at the lines – solid, dashed?

Exercise #5: Software Testing - Test Case Selection

- Using the Low-level requirements in the "Create Low-Level Requirements" exercise from Module 6, create a:
  - Normal range test case
  - Robustness test case



"Software testing is used to demonstrate that the software satisfies its requirements and to demonstrate with a high degree of confidence that errors that could lead to unacceptable failure conditions, as determined by the *system* safety assessment process, have been removed" (Section 6.4)



Hardware/software integration testing: To verify correct operation of the software in the target computer environment.

Software integration testing: To verify the interrelationships between software requirements and components and to verify the implementation of the software requirements and software components within the software architecture.

Low-level testing: To verify the implementation of Low-level requirements.

Executable Object Code complies with the High-level requirements

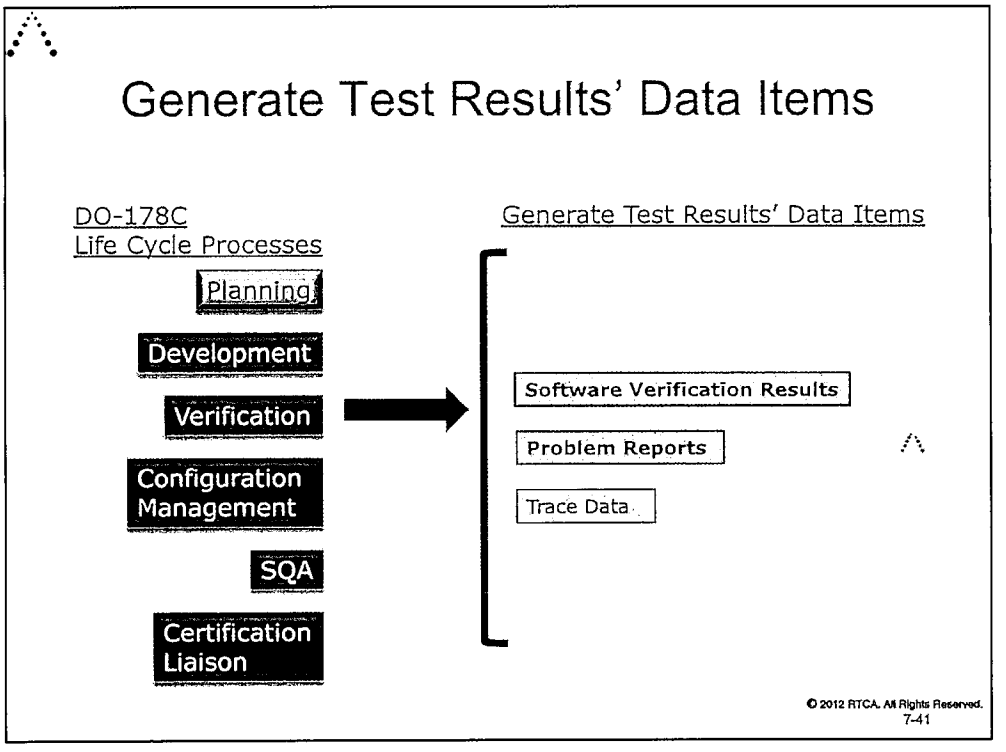
Executable Object Code is robust with the High-level requirements

Executable Object Code complies with the Low-level requirements

Executable Object Code is robust with the Low-level requirements

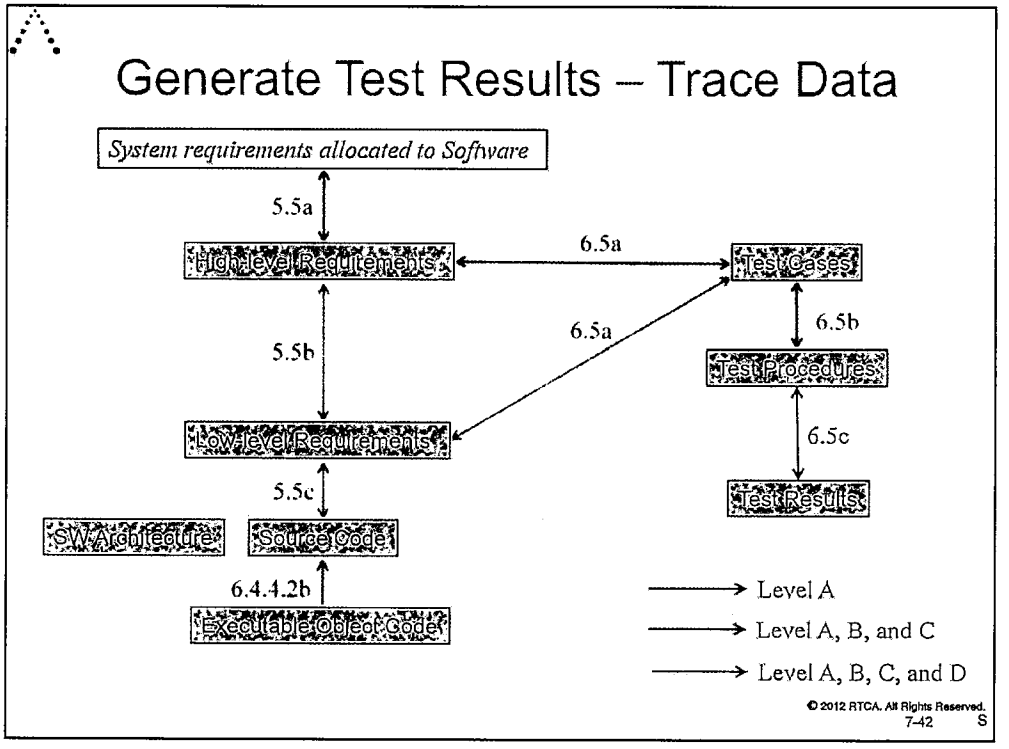
Executable Object Code is compatible with the target computer

For example, to show that an Executable Object Code complies with High-level requirements, you create a test case, create a test procedure and then run the test to get the test's results. Then you determine whether the test results achieved the expected values documented in the test case.



See Second Screen slides titled:

- Software Verification Results
- Problem Reports
- Trace Data



The following trace data is best created during generation of test results.

Again note that identification names may be used as trace data if naming conventions allow.

Verification of Parameter Data Item File

- The Executable Object Code and the Parameter Data Item File should be verified together

Can a Parameter Data Item File be verified separately from the Executable Object Code?

- Yes, if certain criteria are met.

Section 6.6

7-43

Verification of Parameter Data Item Files is done with the verification of the Executable Object Code. This means - All allowable Parameter Data Items Files were used when verifying the Executable Object Code.

See the Glossary for the definition of Parameter data item and Parameter Data Item File.

For more information on Parameter Data Item File see DO-248C section 4.20.

Verification of Parameter Data Item File

- The Executable Object Code and the Parameter Data Item File may be verified separately if:

"The Executable Object Code has been developed and verified by normal range testing to correctly handle all Parameter Data Item Files that comply with their defined structure and attributes."

"The Executable Object Code is robust with respect to Parameter Data Item Files structures and attributes."

"All behavior of the Executable Object Code resulting from the contents of the Parameter Data Item File can be verified."

"The structure of the life cycle data allows the parameter data item to be managed separately."

Section 6.6

7-44

For parameter data items that can be verified separately from verification of the Executable Object Code, the objectives identified below apply. The objectives below may be achieved by a combination of tests, analyses, and reviews.

- The Parameter Data Item File should be verified to comply with its structure as defined by the High-level requirements; this verification includes ensuring that the Parameter Data Item File does not contain any elements not defined by the High-level requirements. Each data element in the Parameter Data Item File should also be shown to have the correct value, to be consistent with other data elements, and to comply with its attributes as defined by the High-level requirements.

  Note: For certain data elements, their attributes may be the only aspect that needs to be verified. In other cases, the value of the data element may need to be verified.

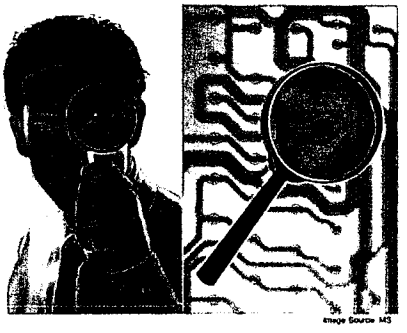
- All elements of the Parameter Data Item File have been covered during verification.

This would generally be done by a review or a tool that would do an analysis of the Parameter Data Item File (A tool would probably need to be qualified)

See the Glossary for the definition of Parameter data item and Parameter Data Item File. For more information on Parameter Data Item File see DO-248C section 4.20.

Does Testing Overlap with Reviews and Analysis?

Class Discussion



What is the difference between reviews and analysis?

Sample answers

A review is a human activity that may give different answers at different times.

An Analysis is a specific set of steps that should always yield the same answer.

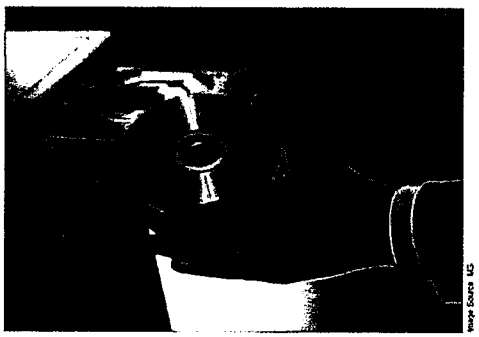
Could testing be considered an analysis?

Yes, but DO-178C requires testing (section 6.4) over and above any reviews and analyses (section 6.3). Testing must be 100% and reviews and analysis are also 100%. One can not replace the other.

End of Part 1

BREAK

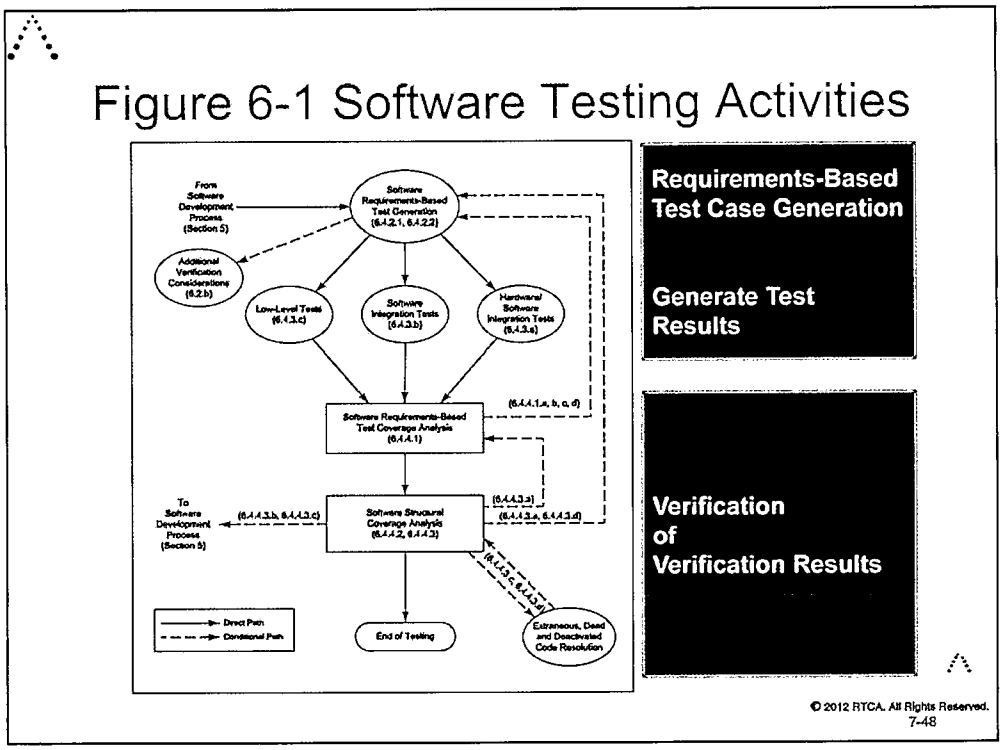
Section
6.4.4
&
6.4.5



Software Verification Results

REVIEWS AND ANALYSIS

Both Reviews and Analysis are required by DO-178C.



This figure is useful to help guide the testing work flow. Work flow may generally go from top to bottom but needs to be specified in the Verification Plan for the specific project.

The top half of the figure illustrates test case generation and testing. The bottom half is Verification of the Verification results. There are feedback loops from the bottom half to the top half which are used if problems are found.

Go to DO-248C for more on Coverage Analysis.

1) Statement Coverage: test executes every source code line

2) Decision Code Coverage: in an if statement you took the true path and the false path

Is Decision Coverage a subset of Statement Coverage?

If you use Source Code, it is. If you use Object Code, that does not that guarantee you did the source code in the object code.

3) Modified Condition / Decision Coverage

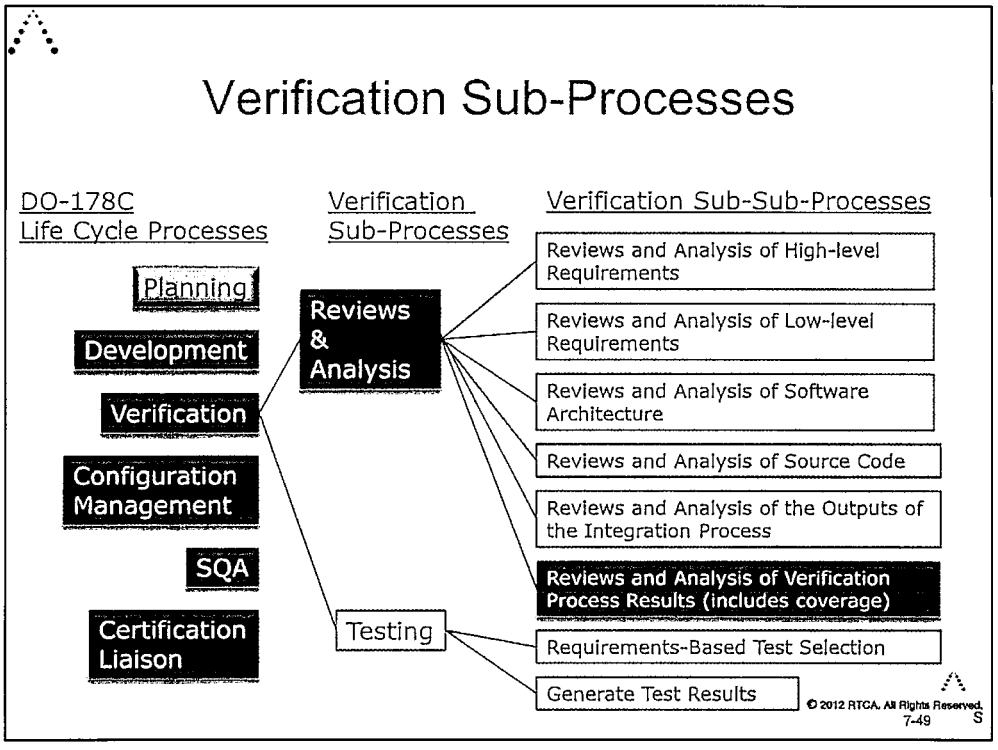
For logic statements, A and B or not C

How many test are needed for this logical expression

x=(a and b) or C

If X

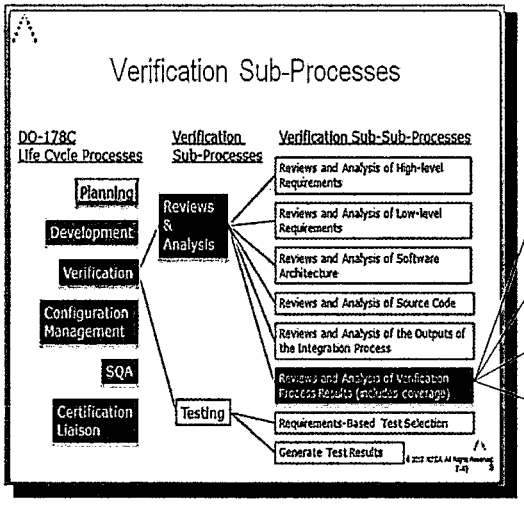
still requires MC/DC



One box we have not yet covered:

- Reviews and Analysis of Verification Process Results (includes coverage)

Verification Sub-Sub-Sub-Processes



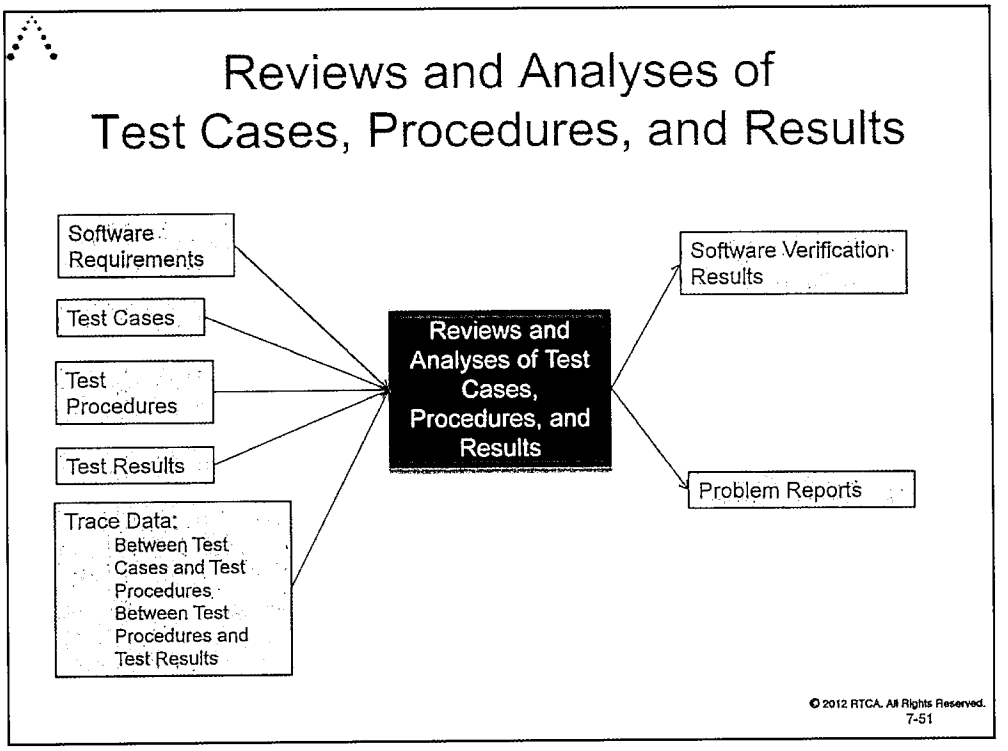
Reviews and Analyses of Test Cases, Procedures, and Results

Test Coverage of High-level and Low-level Requirements

Test Coverage of Software Structure

Test Coverage of Data and Control Coupling

This box breaks up into four parts

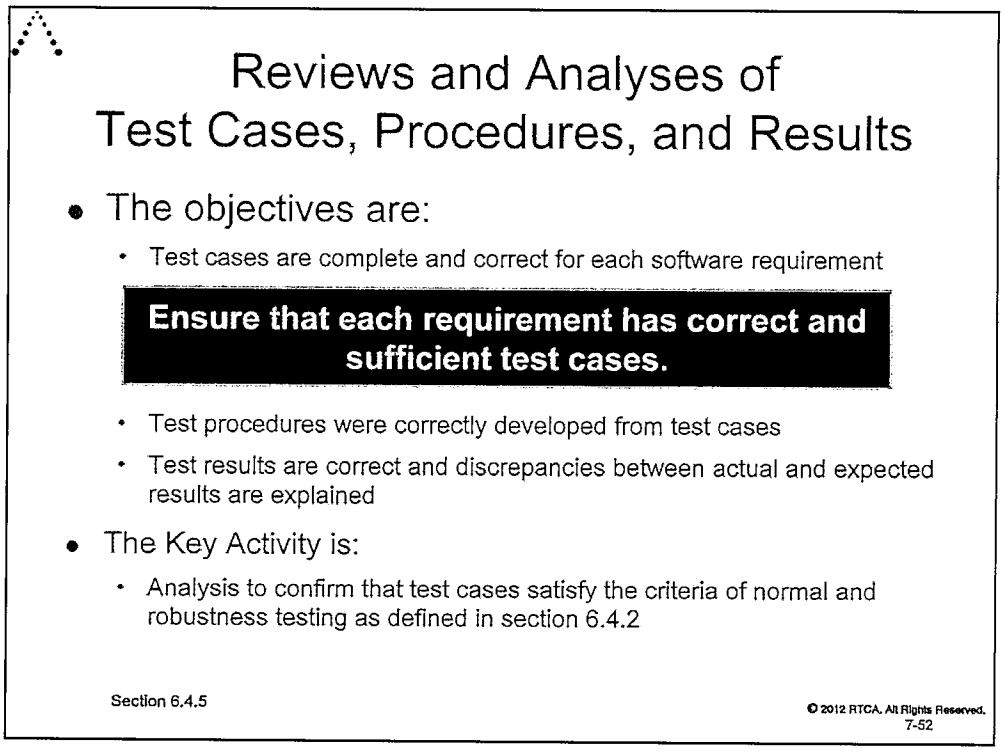


The review or analysis should be done following the documented process in the verification plan.

Software Requirements include both High-level and Low-level requirements

High-level Requirements includes Derived High-level Requirements

Low-level Requirements includes Derived Low-level Requirements



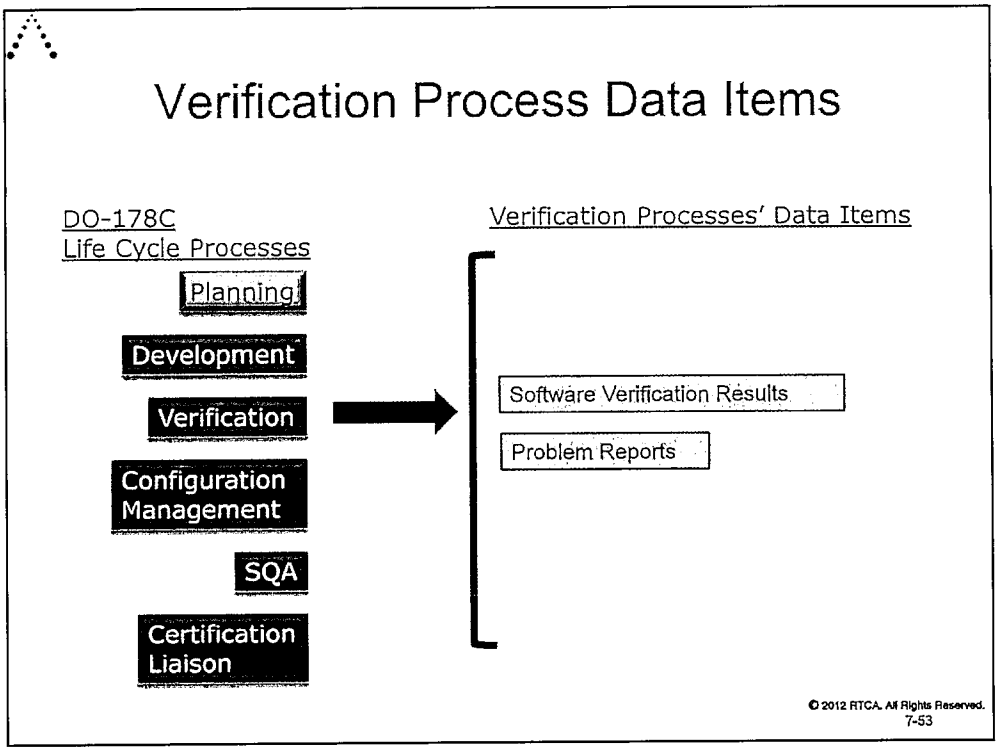
From 6.4.5

- Test cases: The objectives related to verification of test cases are presented in sections 6.4.4.a and 6.4.4.b.
    - From 6.4.4.1b - Analysis to confirm that test cases satisfy the criteria of normal and robustness testing as defined in section 6.4.2. (i.e. Normal an robustness test cases were developed)
- Test procedures: The objective is to verify that the test cases, including expected results, were correctly developed into test procedures.
- Test results: The objective is to ensure that the test results are correct and that discrepancies between actual and expected results are explained.

From 6.4.4a - Test coverage of High-level requirements is achieved.

From 6.4.4b - Test coverage of Low-level requirements is achieved.

Key activity is from section 6.4.4.1b



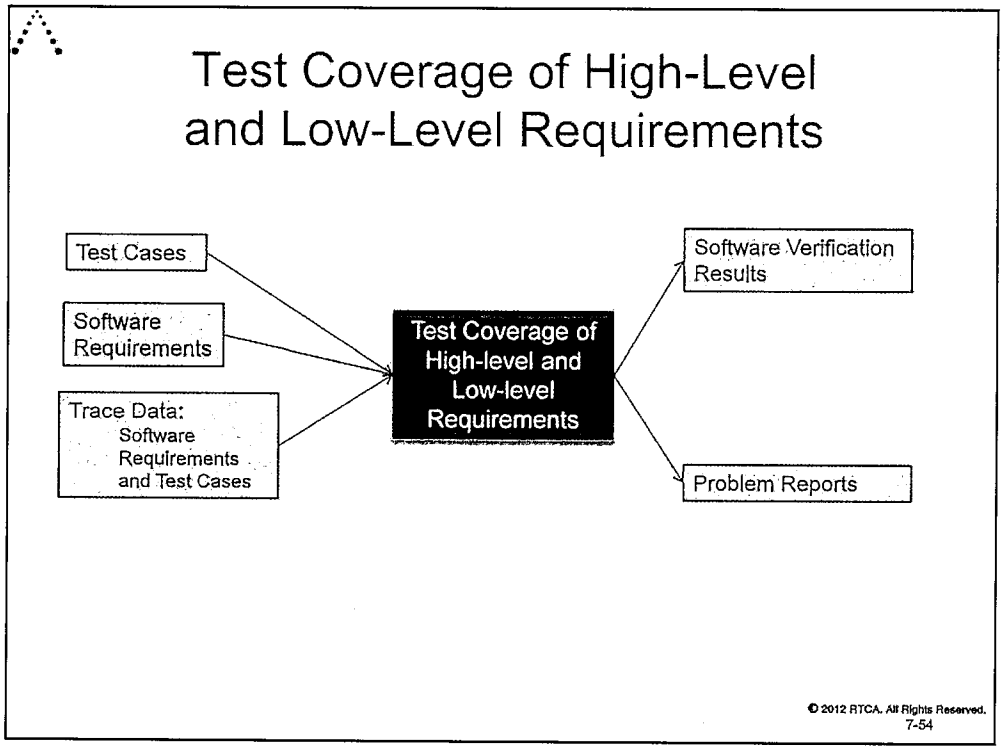
See Second Screen slide titled:

- Software Verification Results

Problem reports data Item is covered in a later slide

See slide near end of this module titled:

- Problem Reports

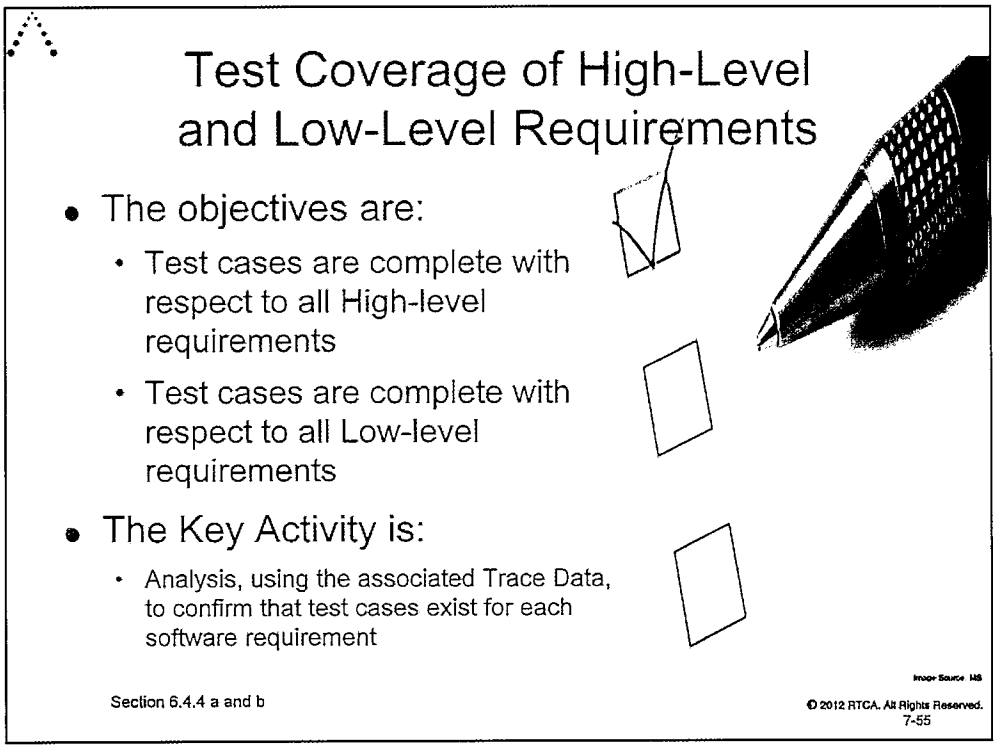


The reviews or analysis should be done following the documented process in the verification plan.

Software Requirements include both High-level and Low-level requirements

High-level Requirements includes Derived High-level Requirements

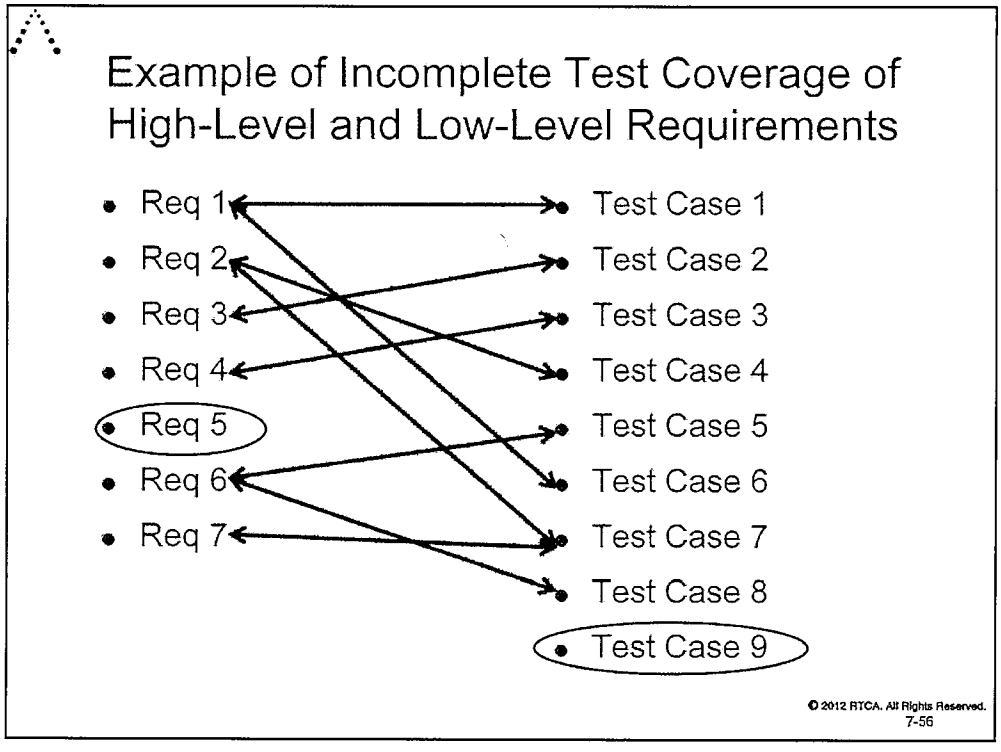
Low-level Requirements includes Derived Low-level Requirements



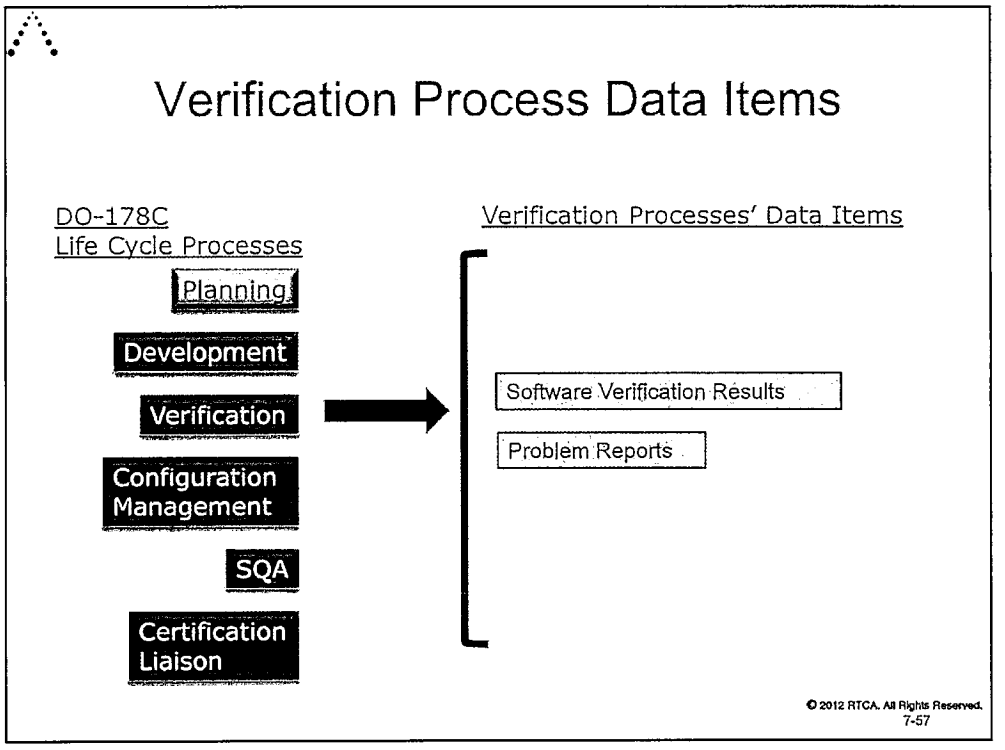
Section 6.4.4a - Test coverage of High-level requirements is achieved.

Section 6.4.4b - Test coverage of Low-level requirements is achieved.

The key activity is from section 6.4.4.1a.



This chart shows that a Requirement (5) does not have a test case, and a Test Case (9) does not have a requirement. There are no expected results. It may drive you to create a requirement (maybe a derived requirement).



See Second Screen slide titled:

- Software Verification Results

Problem reports data Item is covered in a later slide

See slide near end of this module titled:

- Problem Reports

Test Coverage
Structural Coverage Analysis

- An analysis of what was executed by the requirements-based testing
- Performed on Source Code, Object Code, or Executable Object Code
  - Statement coverage
  - Decision coverage
  - "Modified Condition/Decision Coverage" (MC/DC)
- Data and Control Coupling

Image Source: M3

7-58

This analysis is based on what was executed by the requirements-based testing.

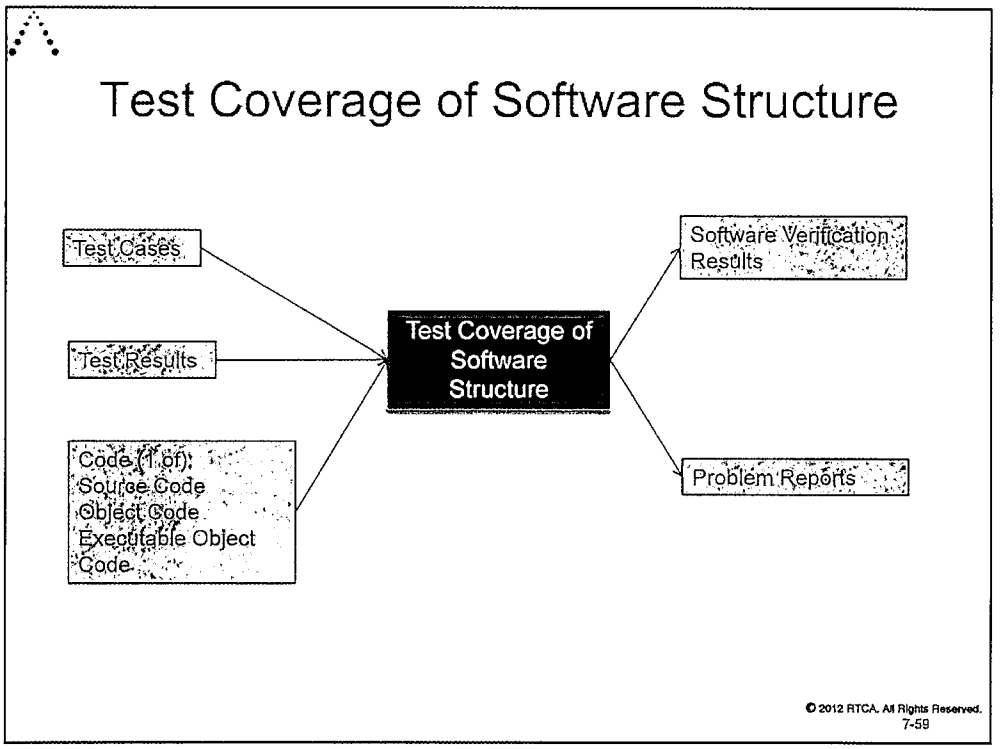
Two types of structural coverage analysis are:

- Software structure
- Data and Control coupling

Note that the software structure can be the structure of the:

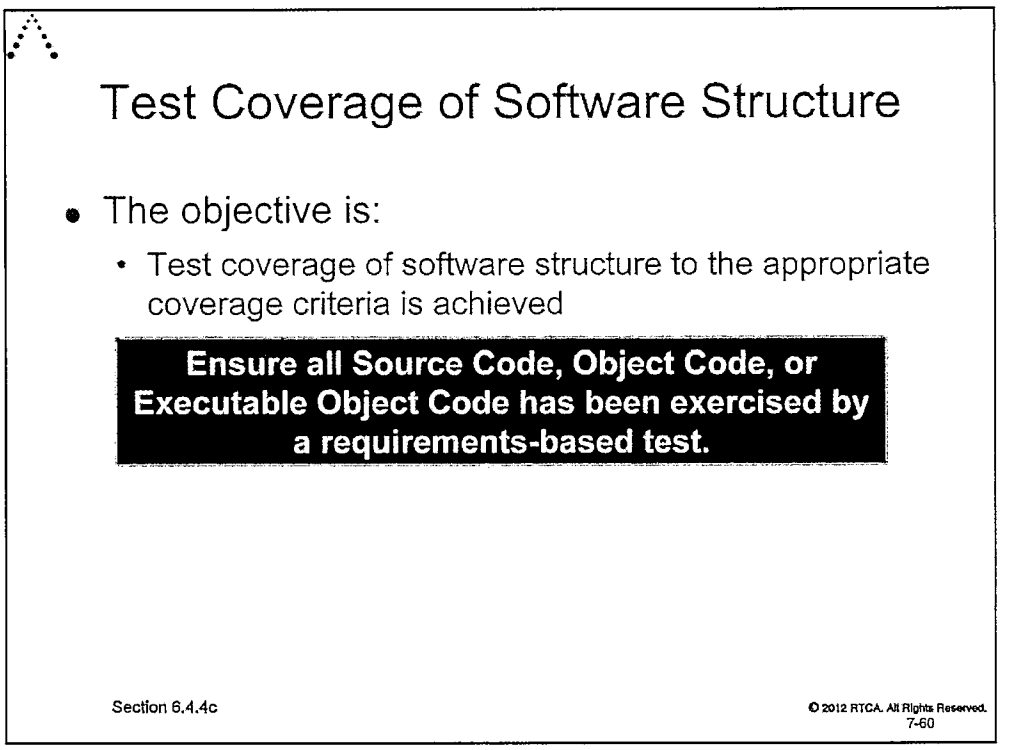
- Source Code
- Object Code
- Executable Object Code

If the analysis is done using the object code or Executable Object Code then DO-248C Section 3.42 provides more information – "FAQ #42: What needs to be considered when performing structural coverage at the object code level?". The structure at an object code level is not necessarily the same as the structure seen at the Source Code level.



The analysis should be done following the documented process in the verification plan.

There are many ways to achieve the objective for this activity.



From DO-248C section 3.43 -

The purpose of structural coverage analysis, along with the associated structural coverage analysis resolution, is to complement requirements-based testing with the following:

- Provide evidence that the code structure was verified to the degree required for the applicable software level.
- Provide a means to support demonstration of absence of unintended functions.
- Establish the thoroughness of requirements-based testing.

Appropriate Coverage Criteria

- Level A – Modified Condition/Decision Coverage (MC/DC)
- Level B – Decision Coverage
- Level C – Statement Coverage

For comprehensive information on each of these see DO-248C Section 4.13.

Note That the term "Path Testing" is not used in DO-178C.

Note that DO-178C does not use the term or imply the use of "Structural Testing" – Not requirements-based tests

How do you deal with dead code? Do you write a test? See next slide for dead code resolution

What is unreachable code? Its dead code, you made a flawed in implementation.

Test Coverage
Structural Coverage Analysis Resolution

- Incorrect or incomplete test cases
- Inadequate Software Requirements
- Extraneous code, including dead code
- Deactivated code

7-61

What should be done if the structural coverage analysis indicates less than 100% coverage?

Section 6.4.4.3

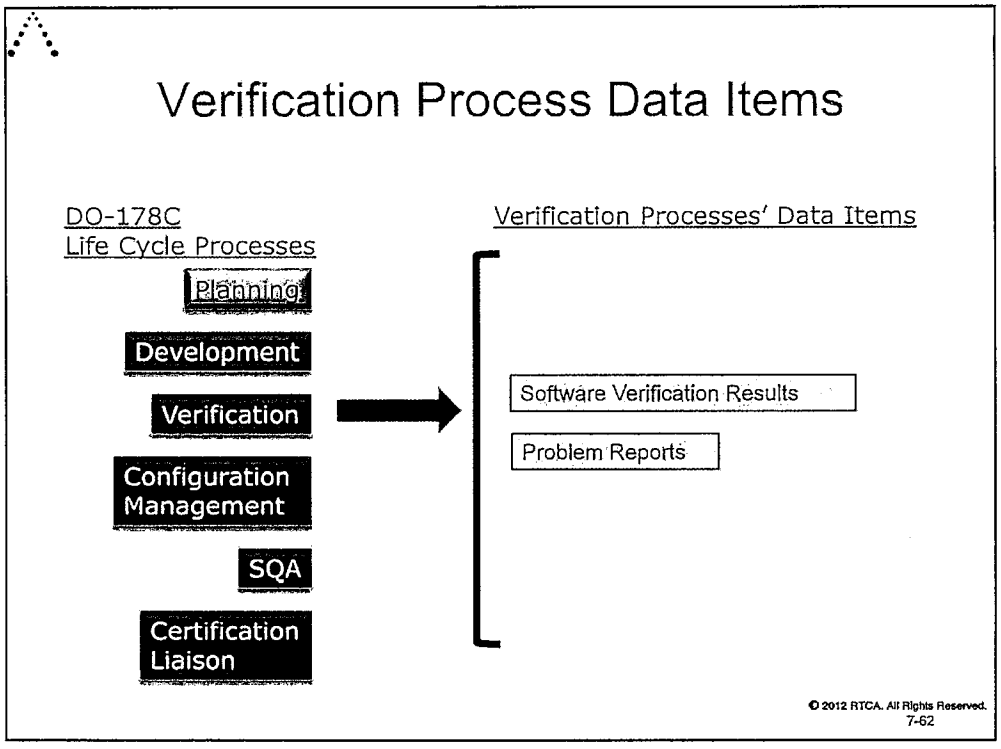
Shortcomings in requirements-based test cases or procedures: The test cases should be augmented or test procedures changed to provide the missing coverage.

Inadequacies in software requirements: The software requirements should be modified and additional test cases developed and test procedures executed.

Extraneous code, including dead code: The code should be removed and an analysis performed to assess the effect and the need for reverification. If extraneous code is found at the Source Code or object code level, it may be allowed to remain only if analysis shows it does not exist in the Executable Object Code (for example, due to smart compiling, linking, or some other mechanism), and procedures are in place to prevent inclusion in future builds.

Deactivated code: Deactivated code should be handled in one of two ways, depending upon its defined category:

- Category One: Deactivated code that is not intended to be executed in any current configuration used within any certified product. For this category, a combination of analysis and testing should show that the means by which the deactivated code could be inadvertently executed are prevented, isolated, or eliminated. Any reassignment of the software level for Category One deactivated code should be justified by the *system* safety assessment process and documented in the Plan for Software Aspects of Certification. Similarly, any alleviation of the software verification process for Category One deactivated code should be justified by the software development processes and documented in the Plan for Software Aspects of Certification.
- Category Two: Deactivated code that is only executed in certain approved configurations of the target computer environment. The operational configuration needed for normal execution of this code should be established and additional test cases and test procedures developed to satisfy the required coverage objectives.



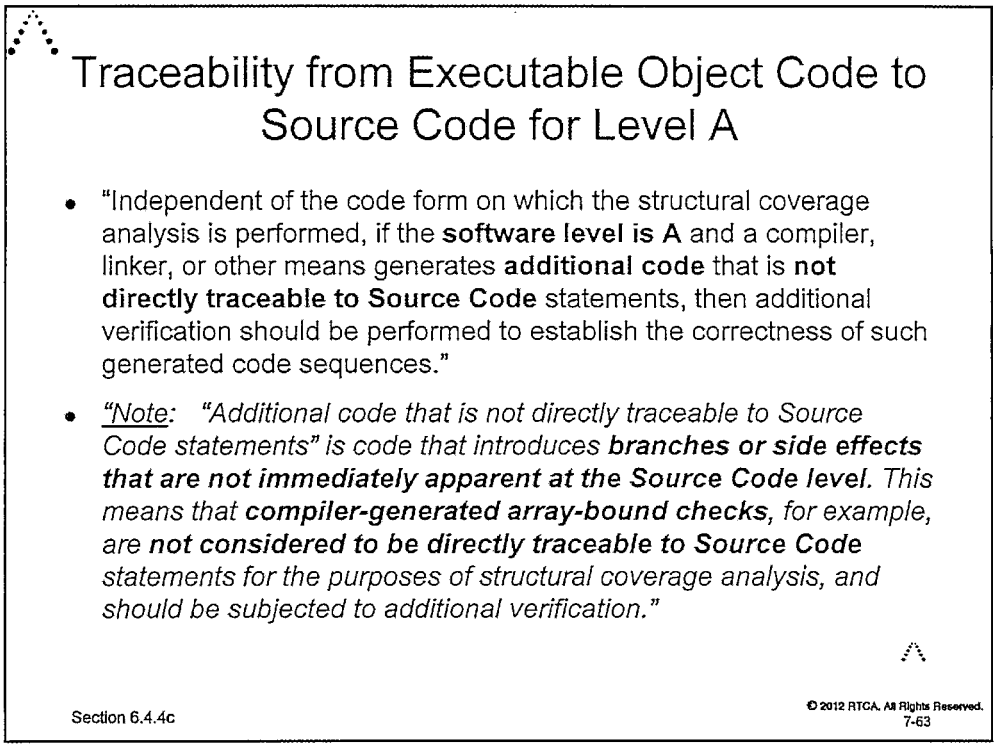
See Second Screen slide titled:

- Software Verification Results

Problem reports data Item is covered in a later slide

See slide near end of this module titled:

- Problem Reports

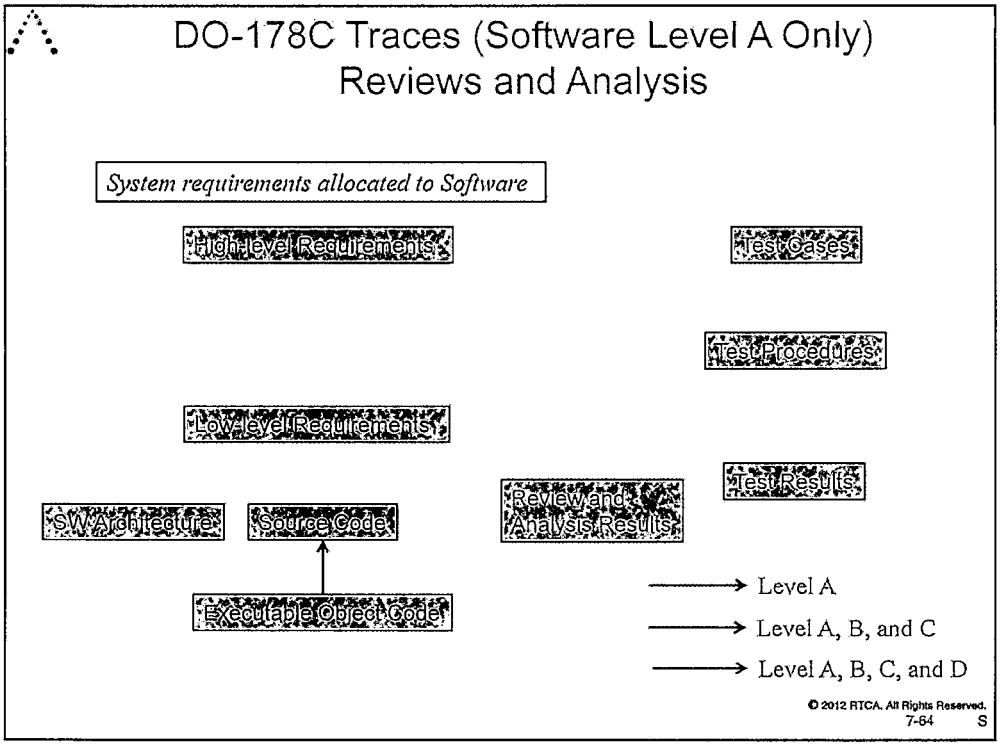


This applies to Level A Software only.

This is a new objective for Annex A Table 7. It was previously called a "hidden objective" as it was in section 6.4.4 but not in the Annex A tables.

The purpose of the object code to Source Code traceability analysis is to detect any functionality added by the compiler, linker, or loader.

See DO-248C section 4.12 for additional information.



This chart shows the trace data needed for the Executable Object Code to Source Code Traceability – Level A Software only.

Test Coverage
Structural Coverage Analysis

- An analysis of what was executed by the requirements-based testing
  - Performed on Source Code, Object Code, or Executable Object Code
    - Statement coverage
    - Decision coverage
    - "Modified Condition/Decision Coverage" (MC/DC)
- Data and Control Coupling

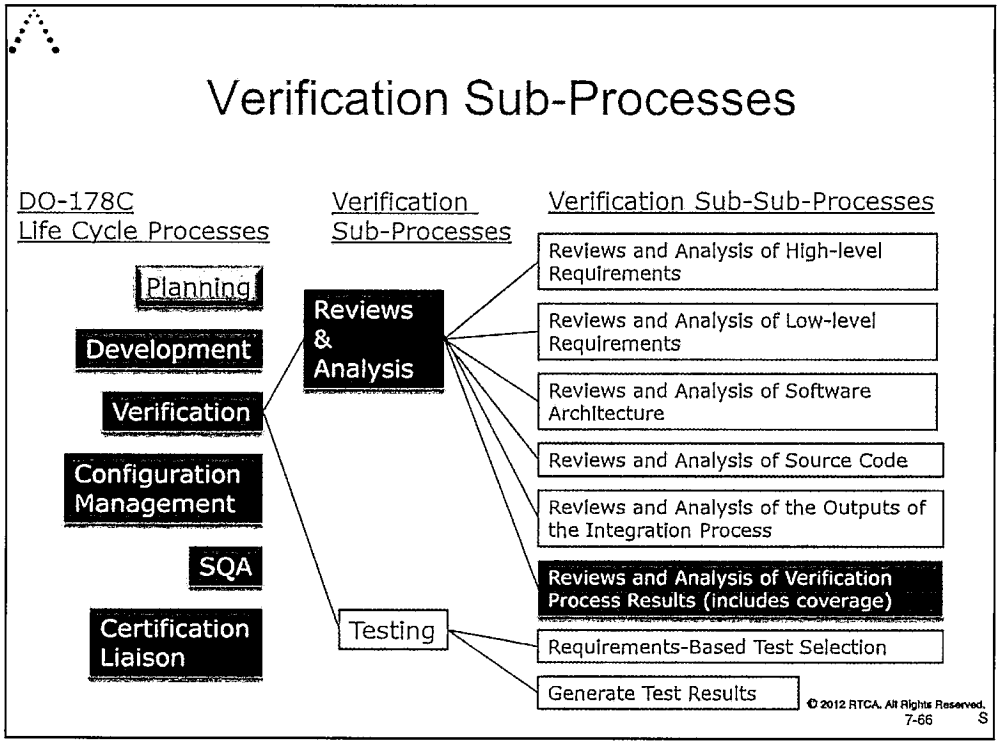
Image Source: MS

7-65

Structural coverage analysis includes coverage by requirements-based testing of data and control coupling.

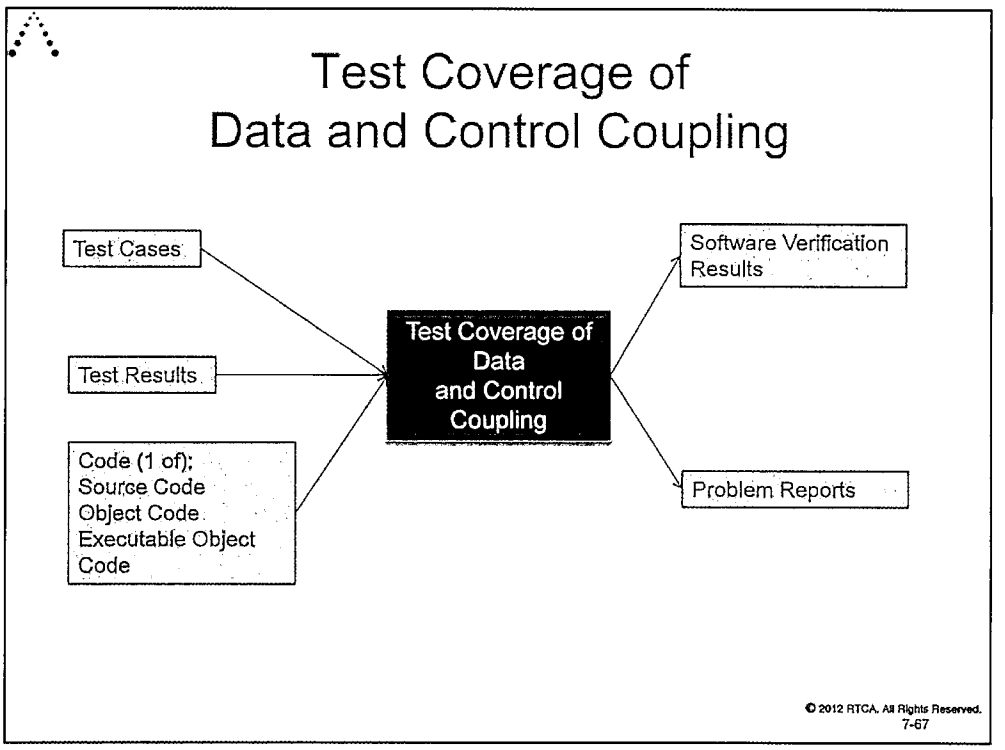
More information may be found in DO-248C

- FAQ #32: What are defensive programming practices?
- FAQ #67: What is analysis of data coupling and control coupling?
- FAQ #80: What needs to be considered when using inlining?
- FAQ #82: If pseudocode is used as part of the Low-level requirements, what issues need to be addressed?

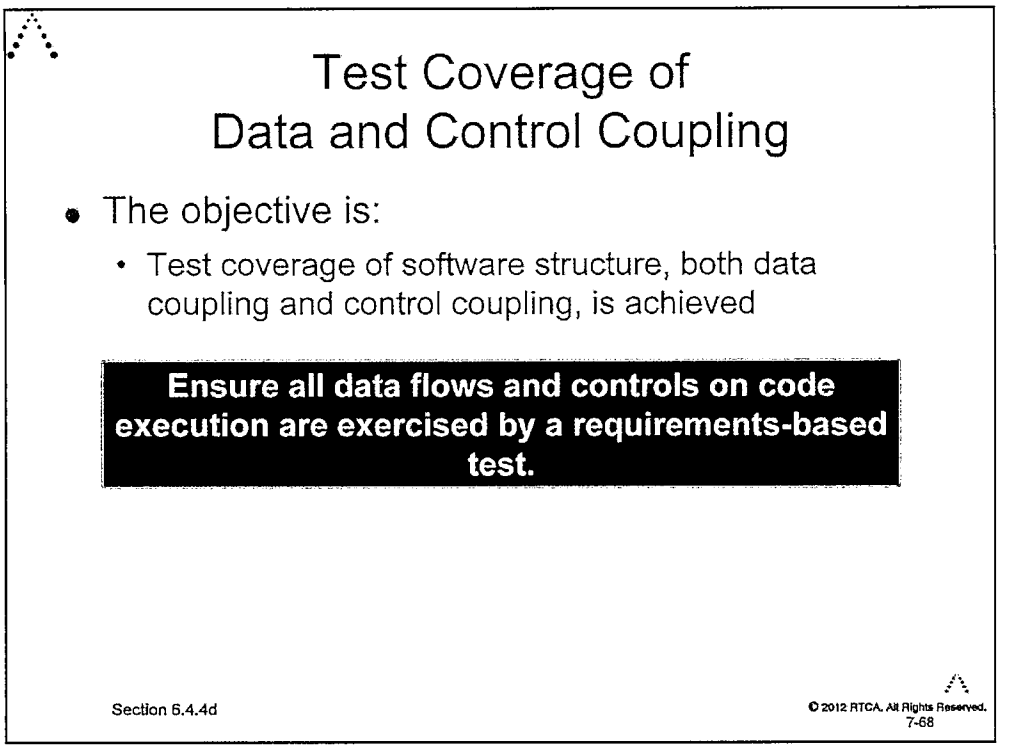


One box we have not yet covered:

- Reviews and Analysis of Verification Process Results (includes coverage)



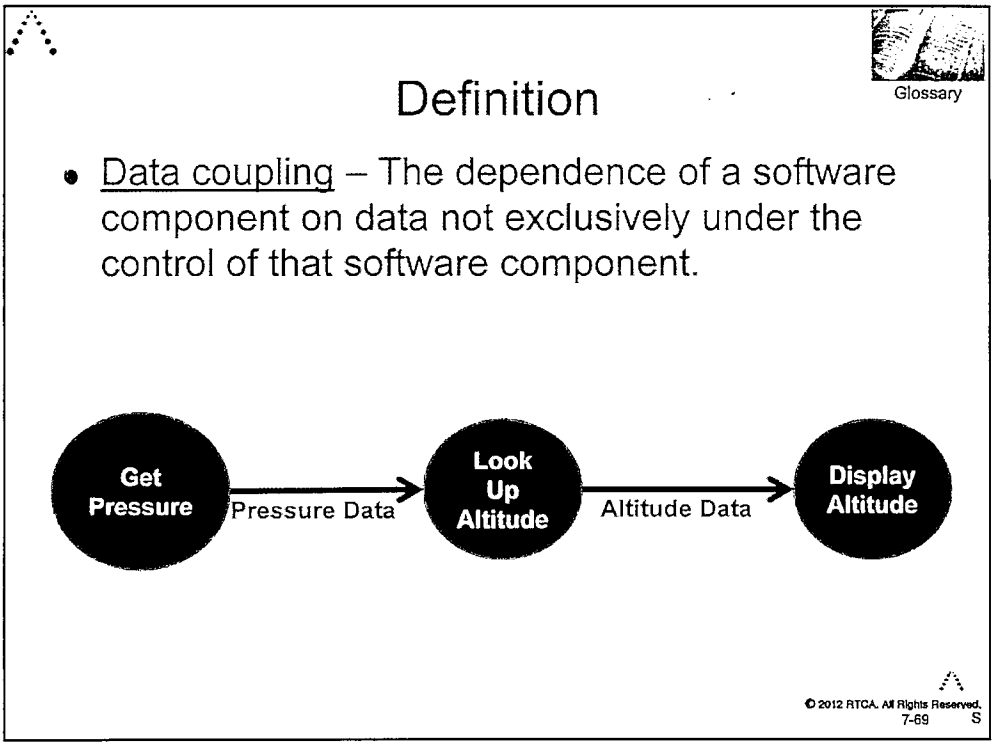
The analysis should be done following the documented process in the verification plan.



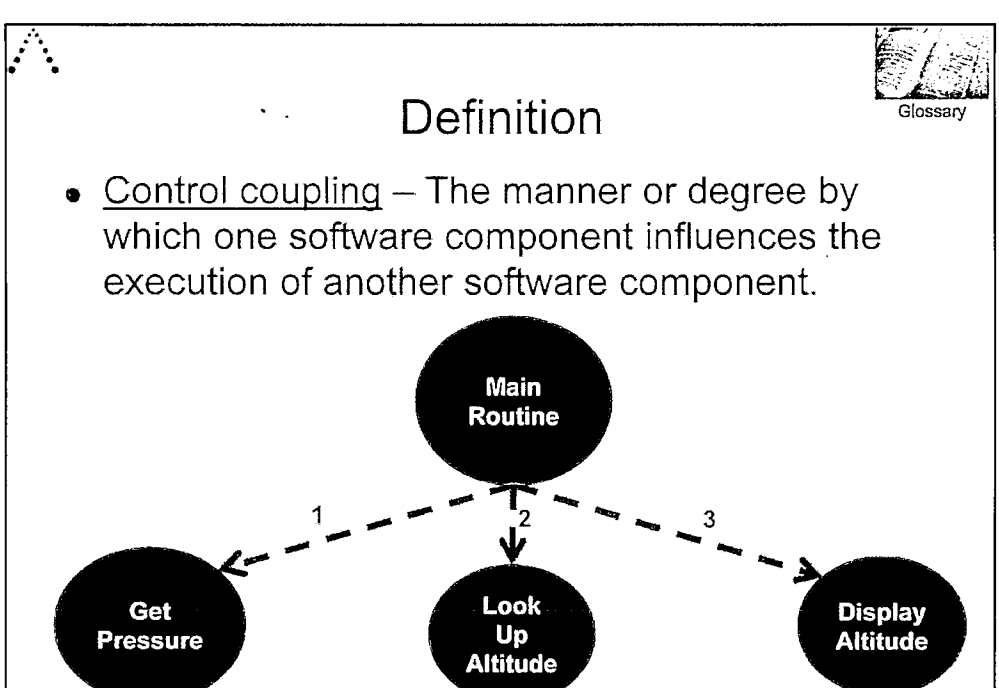
The intent is to ensure that a sufficient amount of hardware/software integration testing and/or software integration testing is performed to:

- Verify that the required data gets to a Software Function
- Verify that the software functions are invoked in the required order

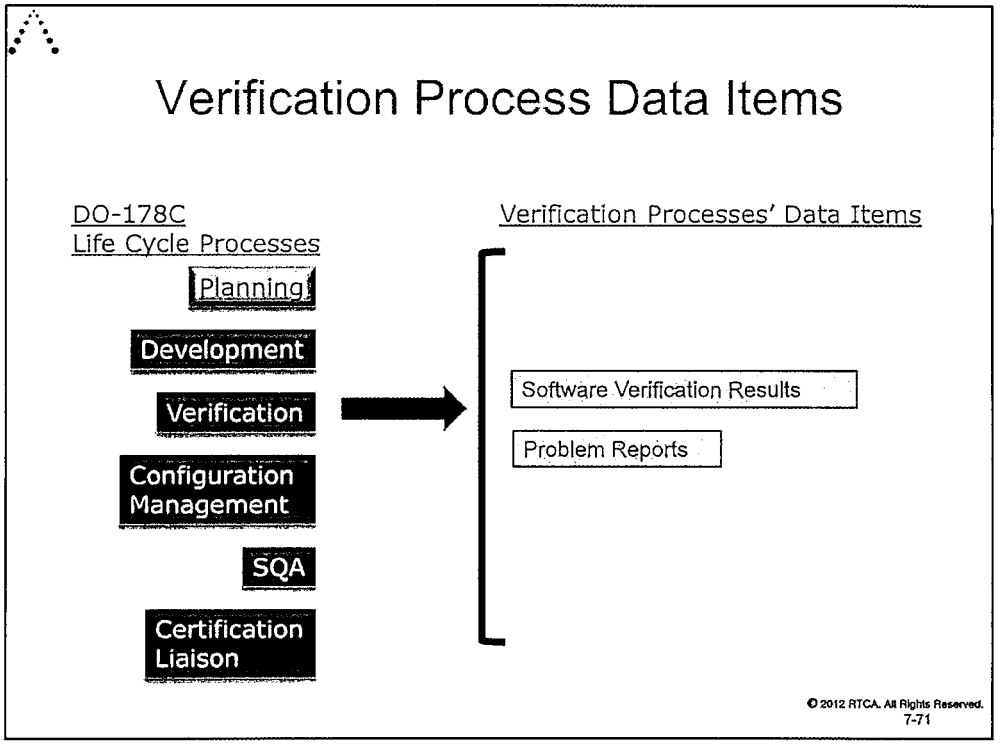
100% coverage is required at Software Levels A, B, and C



For our purposes here : Processes are made up of sub-processes. Sub-processes are made up activities. Another way to look at this is: Processes are made up of processes and/or activities (a recursive definition). We will encounter some sub-sub-processes.



For our purposes here: Processes are made up of sub-processes. Sub-processes are made up activities. Another way to look at this is: Processes are made up of processes and/or activities (a recursive definition). We will encounter some sub-sub-processes.



See Second Screen slide titled:

- Software Verification Results

Problem reports data Item is covered in a later slide

See slide near end of this module titled:

- Problem Reports

Test Coverage
Structural Coverage Analysis Resolution

- Incorrect or incomplete test cases
- Inadequate Software Requirements
- Extraneous code, including dead code
- Deactivated code

Image Source: M3

7-72

What should be done if the structural coverage analysis indicates less than 100% coverage?

Section 6.4.4.3

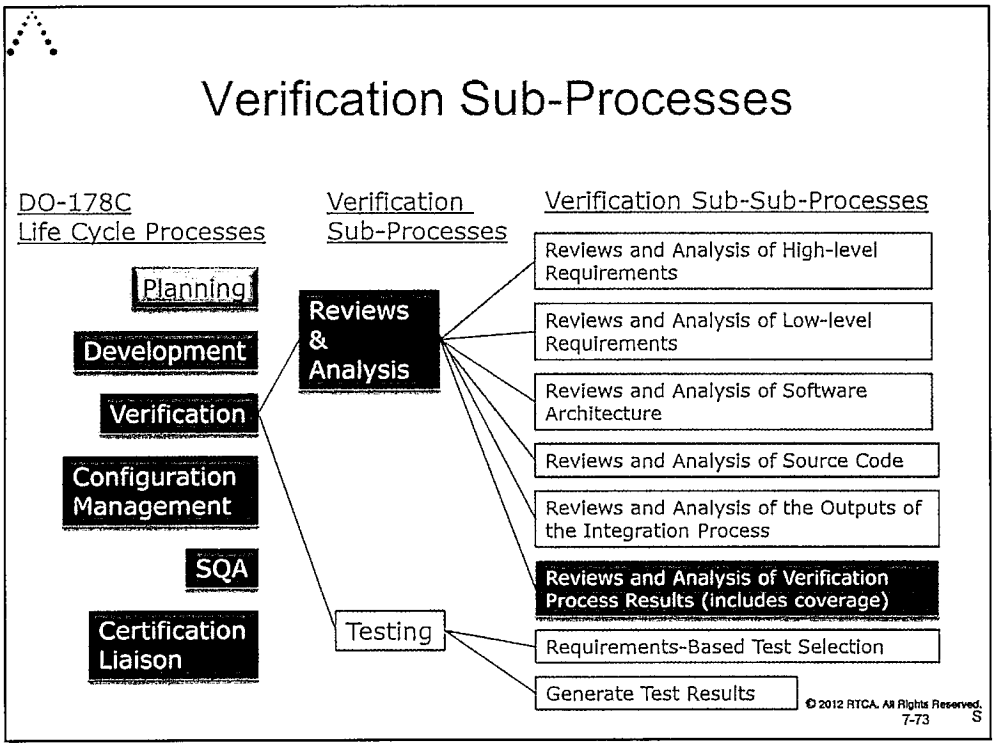
Shortcomings in requirements-based test cases or procedures: The test cases should be augmented or test procedures changed to provide the missing coverage.

Inadequacies in software requirements: The software requirements should be modified and additional test cases developed and test procedures executed.

Extraneous code, including dead code: The code should be removed and an analysis performed to assess the effect and the need for reverification. If extraneous code is found at the Source Code or object code level, it may be allowed to remain only if analysis shows it does not exist in the Executable Object Code (for example, due to smart compiling, linking, or some other mechanism), and procedures are in place to prevent inclusion in future builds.

Deactivated code: Deactivated code should be handled in one of two ways, depending upon its defined category: (Note: Categories for Deactivated code are new to DO-178C)

- Category One: Deactivated code that is not intended to be executed in any current configuration used within any certified product. For this category, a combination of analysis and testing should show that the means by which the deactivated code could be inadvertently executed are prevented, isolated, or eliminated. Any reassignment of the software level for Category One deactivated code should be justified by the *system* safety assessment process and documented in the Plan for Software Aspects of Certification. Similarly, any alleviation of the software verification process for Category One deactivated code should be justified by the software development processes and documented in the Plan for Software Aspects of Certification.
- Category Two: Deactivated code that is only executed in certain approved configurations of the target computer environment. The operational configuration needed for normal execution of this code should be established and additional test cases and test procedures developed to satisfy the required coverage objectives.



One box we have not yet covered:

- Reviews and Analysis of Verification Process Results (includes coverage)

Verification of the Verification Results

QUICK REVIEW

Test Coverage Objectives

Image Source: MS

- "Test coverage of High-level requirements is achieved."
- "Test coverage of Low-level requirements is achieved."
- "Test coverage of software structure to the appropriate coverage criteria is achieved."
- "Test coverage of software structure, both data coupling and control coupling, is achieved."

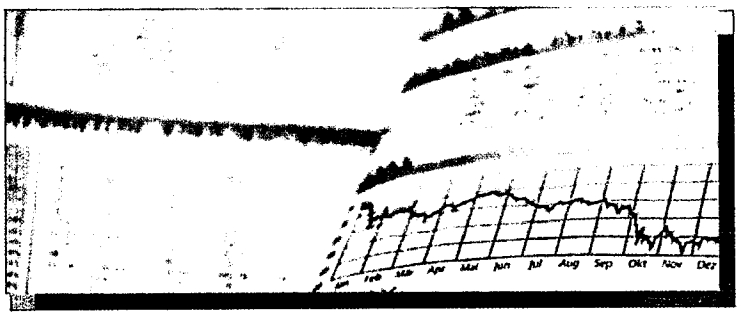
Section 6.4.4

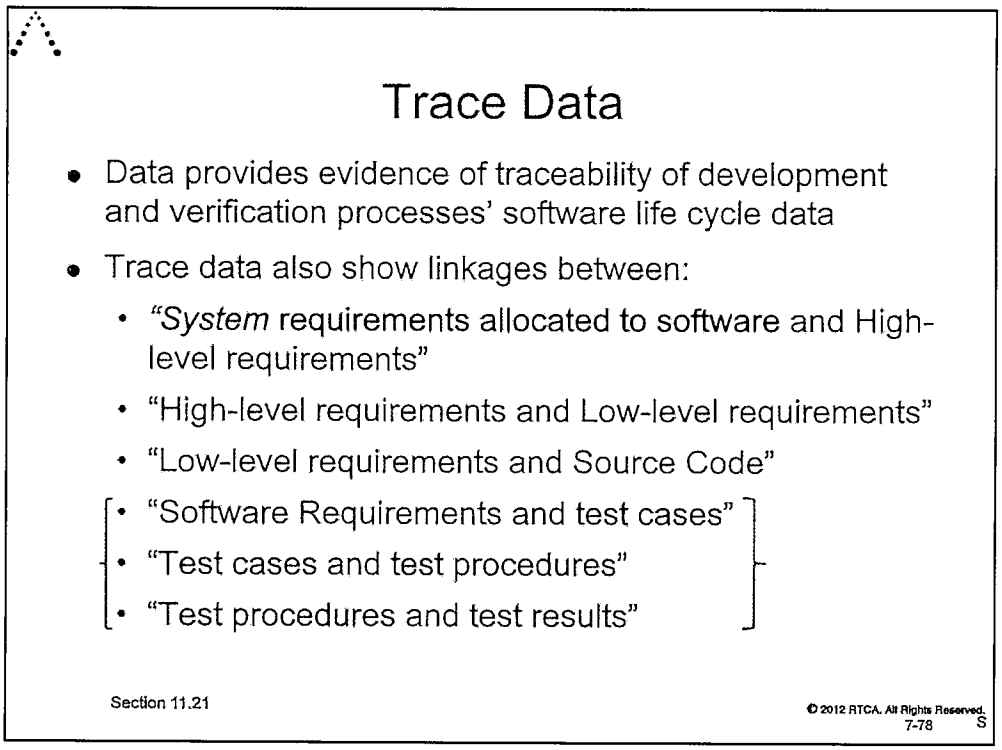
Test Case Reviews and Analysis

- Test cases completely and correctly test the software requirements
- Test procedures are correct with respect to the test cases
- Test results are correct (agree with expected results of the test cases)

Trace Data

- Between software requirements and test cases
- Between test cases and test procedures
- Between test procedures and test results

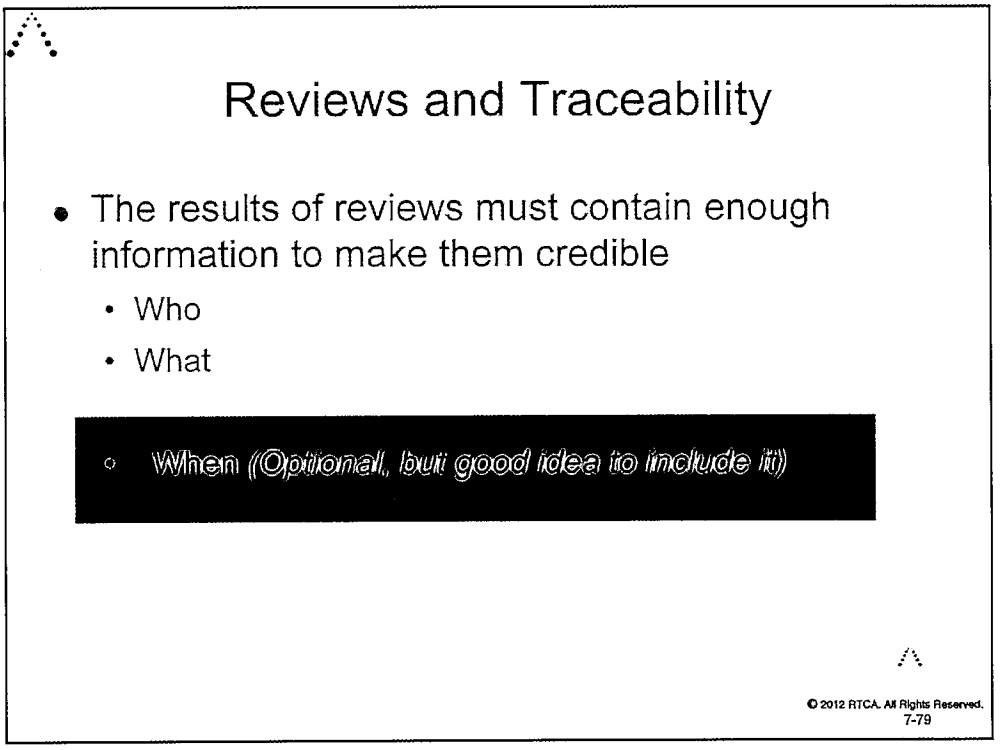




Trace data for this module includes the last three bullets on this slide.

- Software Requirements and test cases
- Test cases and test procedures
- Test procedures and test results

Software requirements include both High-level and Low-level Requirements and their derived requirements.

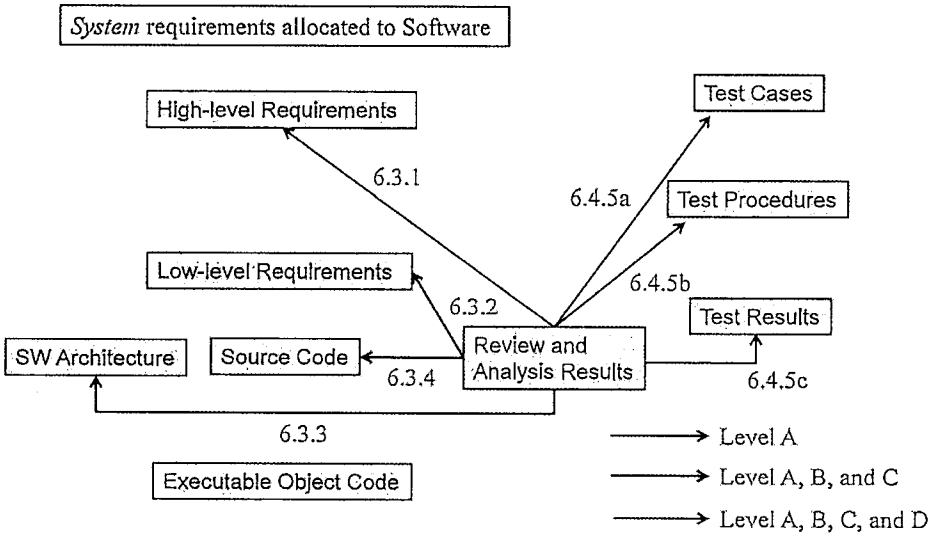


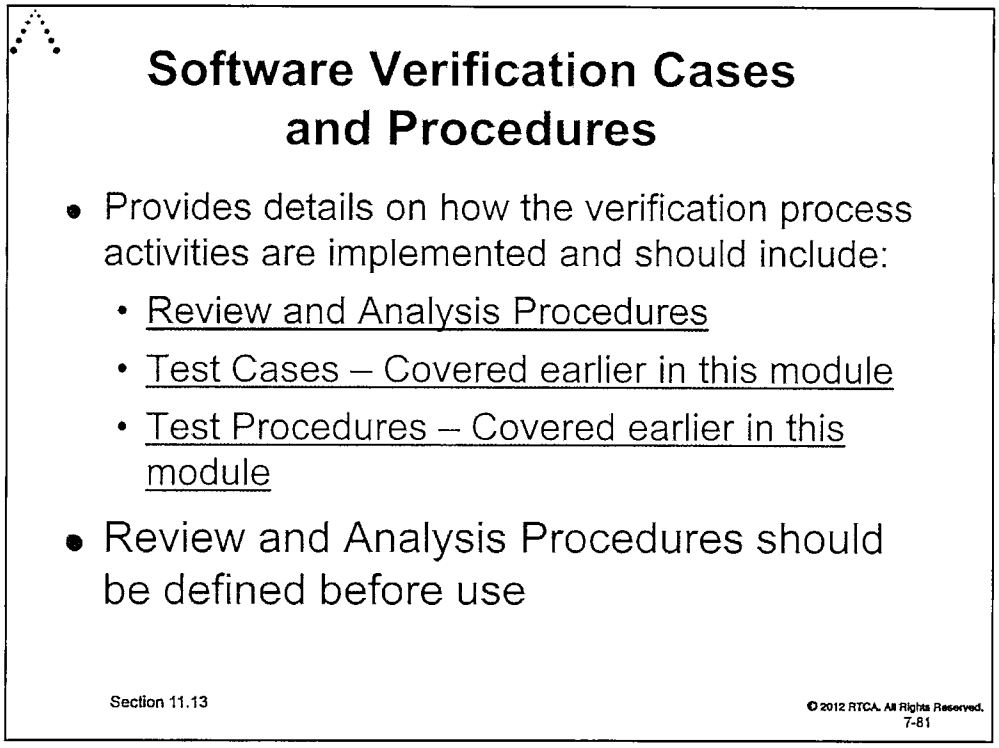
Who did the review? – may be more than one person

What was reviewed? – the Identification of all Software Life Cycle Data reviewed and used during the review

While there is no trace data required by DO-178C it is a good idea to have the reviews pointing to the "What". The next slide indicates this by one-direction arrows.

DO-178C Traces
Reviews and Analysis





Review and Analysis procedures depend on the types of reviews and analysis to be done.

- For reviews it should specify:
  - who should be involved
  - how much prep work is needed
  - which checklists are to be used
  - Whether a meeting or desk check is appropriate
  - etc.
- For Analysis, some things to consider are:
  - Tools involved
  - Mathematical equations to use
  - Step-by-step activities required
  - etc.

Problem Reports

- Identify and record the resolution of software product anomalous behavior, process non-compliance with plans, standards and deficiencies in Software Life Cycle Data
- Reports include:
  - Identification of configuration item and/or life cycle process activity with a problem
  - Identification of configuration item to be modified, or description of process to be changed
  - Detailed problem description for clear understanding, resolution with corrective action

Review Questions (1 of 3)

- What are some of the Software Testing Objectives for the Executable Object Code?

_____

_____

_____

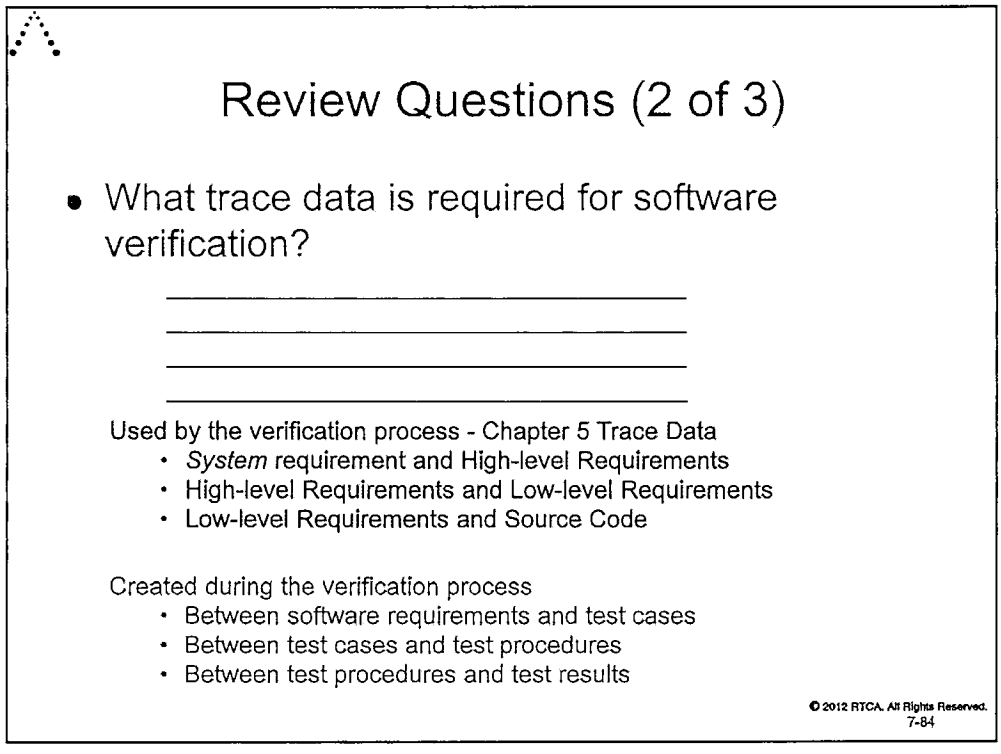
_____

_____

- Complies with the High-level requirements
- Is robust with the High-level requirements
- Complies with the Low-level requirements
- Is robust with the Low-level requirements
- Is compatible with the target computer

Answers:

- Complies with the High-level requirements
- Is robust with the High-level requirements
- Complies with the Low-level requirements
- Is robust with the Low-level requirements
- Is compatible with the target computer

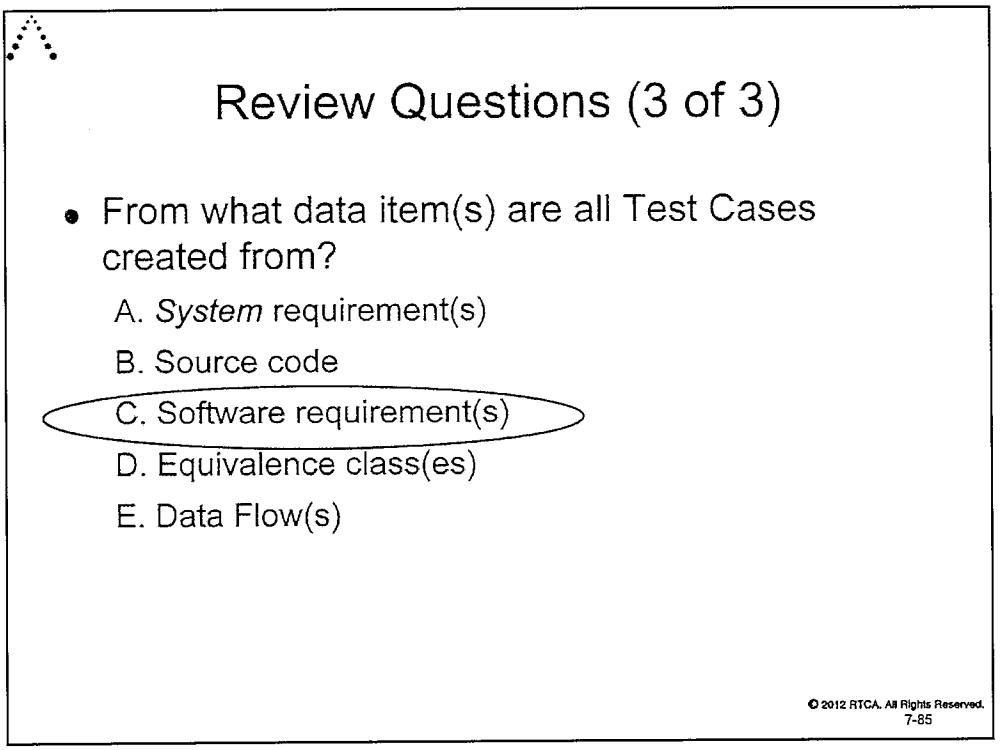


Used by the verification process - Chapter 5 Trace Data

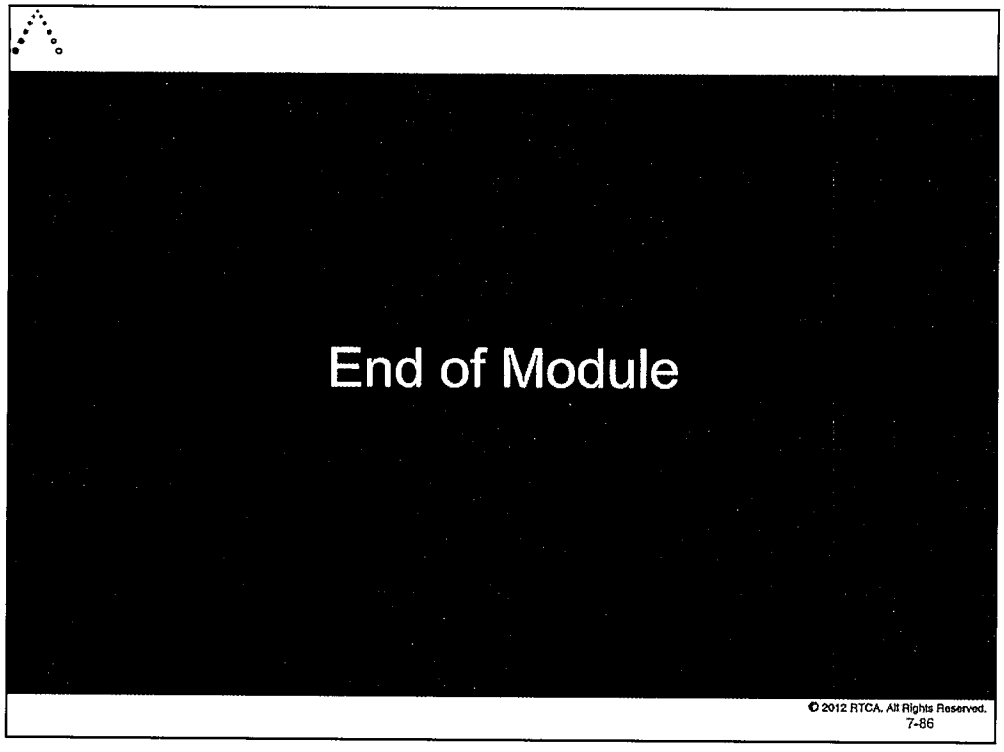
- *System* requirement and High-level Requirements
- High-level Requirements and Low-level Requirements
- Low-level Requirements and Source Code

Created during the verification process

- Between software requirements and test cases
- Between test cases and test procedures
- Between test procedures and test results



Answer is C, Software requirements


End of Module
© 2012 RTCA. All Rights Reserved.
7-86

Exercise # 4
Review Source Code

Do a review of the Instructor-supplied Source Code using the checklist you created in Module 5 – "Code Review Checklist".

Write down any problems you find because we will use them later to create a Problem Report.

Obtain Instructor-supplied Source Code for Table Look Up of Altitude Using Pressure.

Answer Template

Verification Results ID:
Reviewer:
Date:
Source Code ID:

Checklist Item 1	Pass	Fail
:		
:		
Checklist Item n	Pass	Fail
Overall Review Result	Pass	Fail

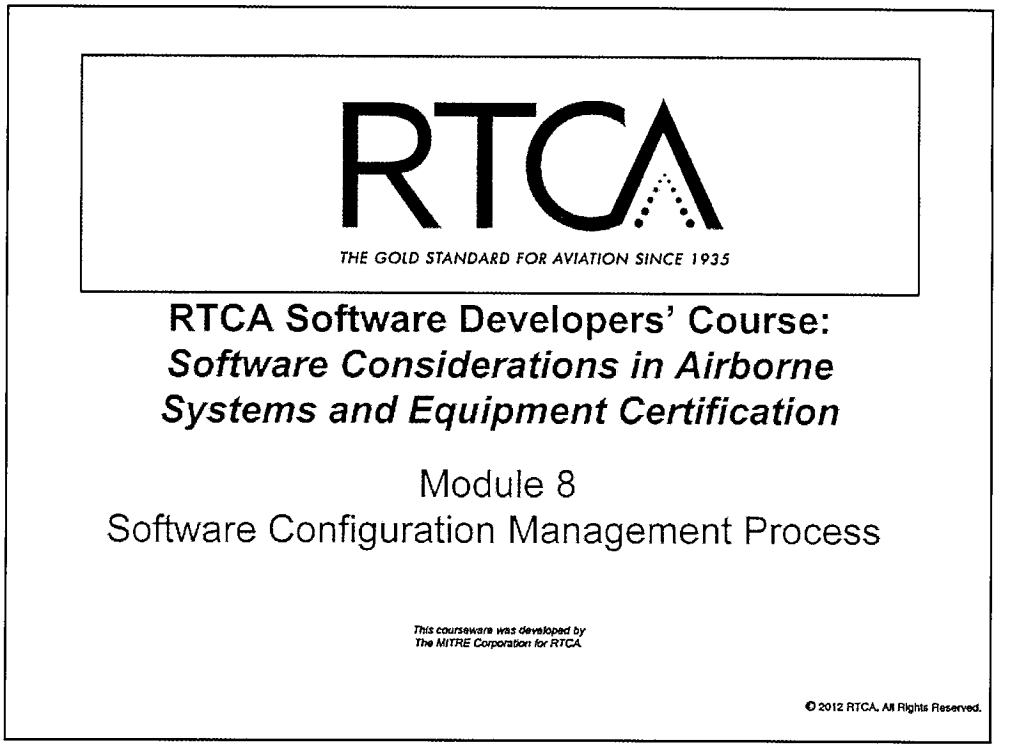
Exercise # 5
Create Test Cases

Using the Low-level requirements in the "Create Low-Level Requirements" exercise from Module 6, create a:

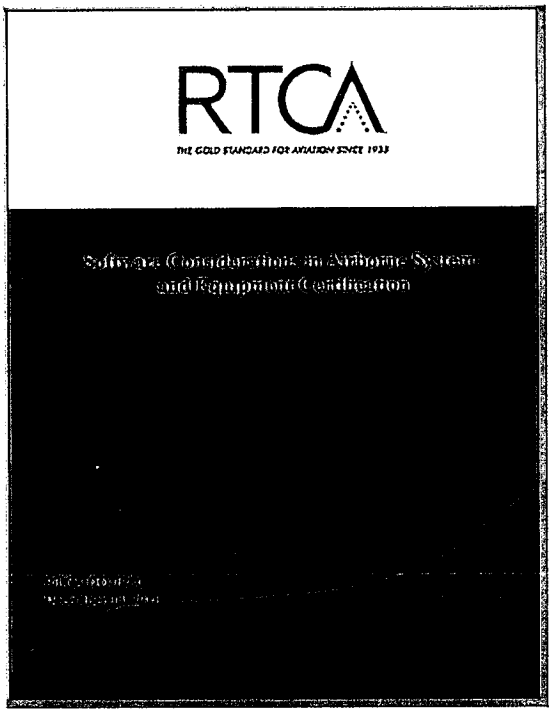
- Normal range test case
- Robustness test case

Test Cases made up of:

Author:
Date:
Low-level Requirements ID:
Inputs:
Expected Outputs:



This 1 hour and 30 minute module introduces the Configuration Management process. For this process, objectives, outputs and activities are described. DO-178C objectives for item identification, change control, archival, releases, problem reporting and baselines are covered. Objectives for different control categories are also studied including control of inputs and outputs during software life cycle that ensures consistency and repeatability of process activities.



Section 7

Module Objectives

At the conclusion of the module, the participant will be able to:

- Describe Software Configuration Management (SCM) as an integral process
- Know that DO-178C Section 7 establishes the objectives, activities, and considerations needed to:
  - Identify and control the inputs and outputs of the software life cycle processes
  - Ensure that the software in the airborne *system* is the software intended for that *system* and is uniquely identified
  - Provide controls that ensure problems receive attention
  - Ensure that the software in the airborne *system* is available in the future if needed

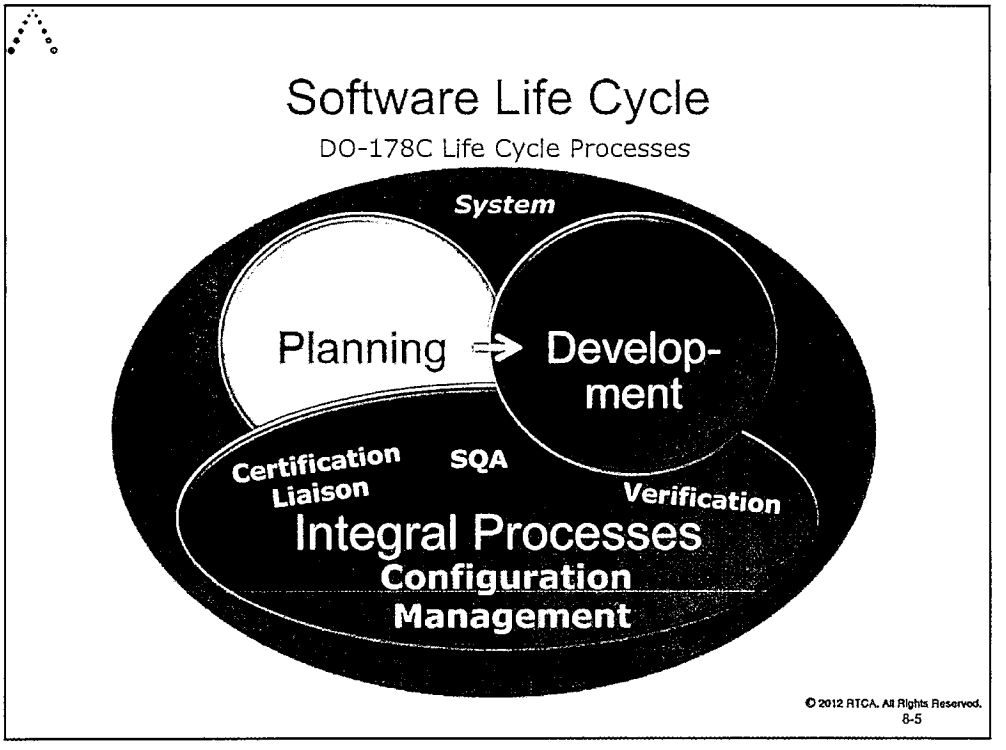
This chart outlines the objectives covered in Module 9.

Module Overview

- SCM Process
- SCM Process Objectives
- Data Control Categories
- SCM Process Activities
- Software Load Control
- Software Life Cycle Environment Control

8-4

This module will cover Software Configuration Management Process, Objectives, activities and controls: Data, Software Load and Software Life Cycle Environment.



There are many life cycle process charts.

This slide depicts the life cycle processes in DO-178C that are used for software.

SCM Process

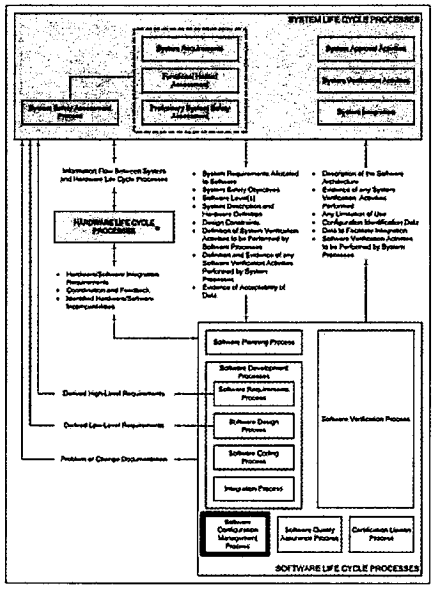
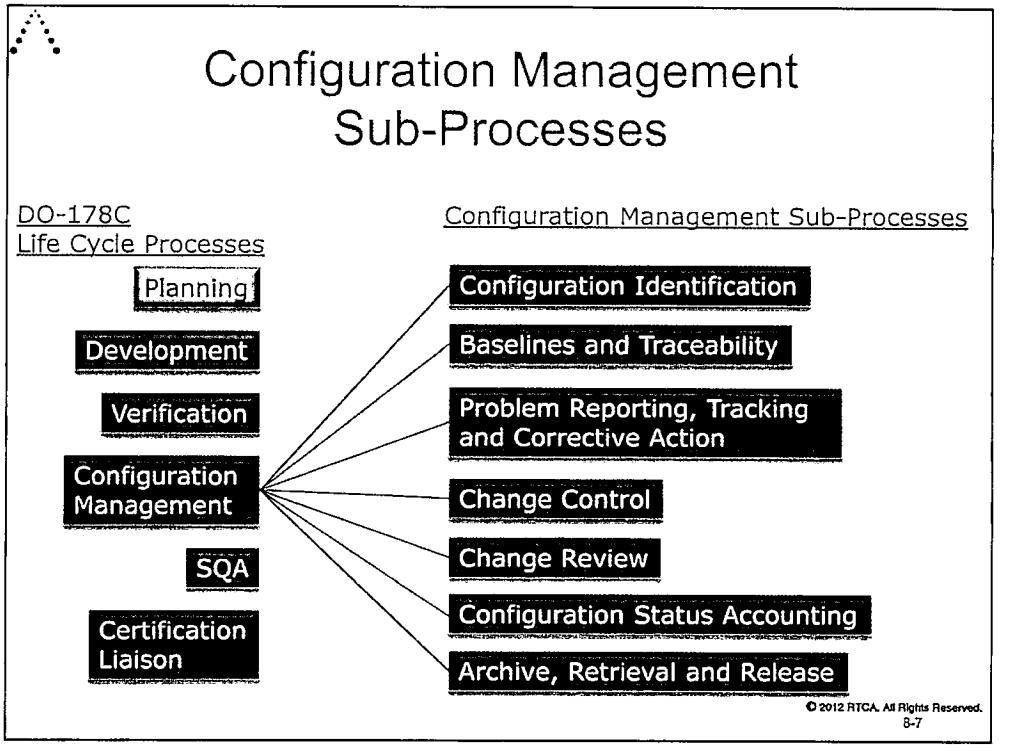


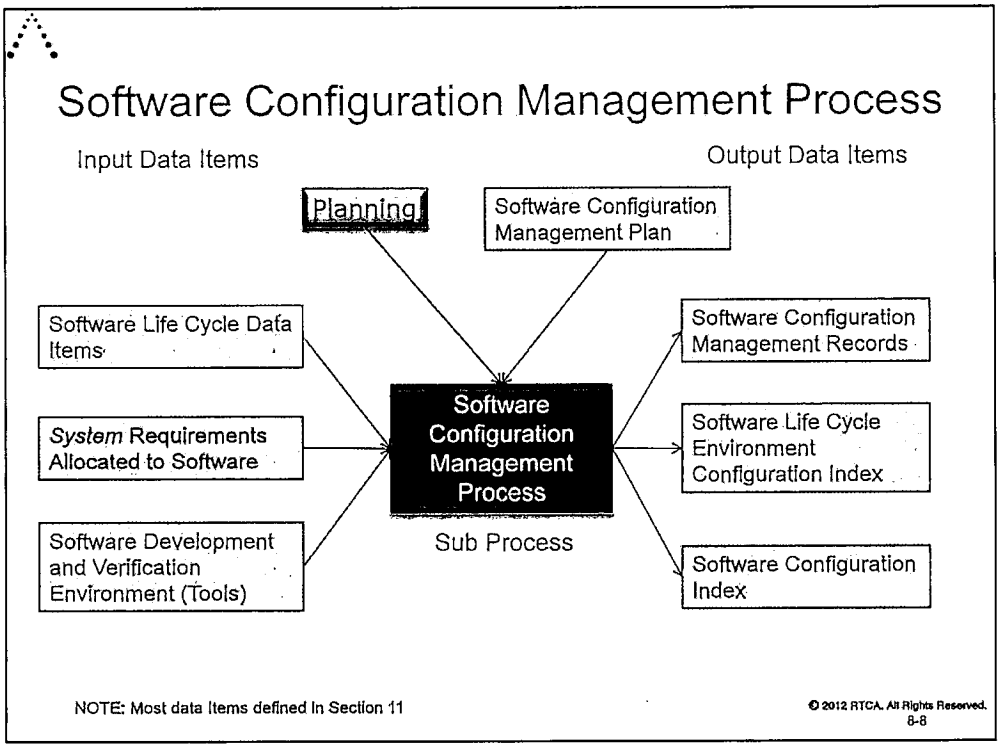
Figure 2-1.

8-6

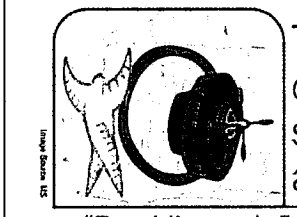
This is a more detailed diagram from DO-178C where we have highlighted the section addressing software configuration management process.



These are the sub-processes associated with Software Configuration Management.



Shown above are input and output data items, and the plan that governs the activities.



The SCM process, working in cooperation with the other software life cycle processes, assists in:

- "Providing a defined and controlled configuration of the software throughout the software life cycle"
- "Providing the ability to consistently replicate the Executable Object Code and Parameter Data Item Files"
- "Providing control of process inputs and outputs during the software life cycle"
- "Providing a known point for review, assessing status, and change control""
- "Providing controls that ensure problems receive attention and changes are recorded, approved, and implemented"
- "Ensuring that secure physical archiving, recovery, and control are maintained for the configuration items"

Section 7

8-9

These are some of the reasons software configuration management is important.

Physical archiving means the ability to get it whenever your plan says.

An interesting point is that DVDs are dying in three or four years. Card readers may have to be obtained from the Smithsonian. Older versions MS Word are no longer useable. How will you retrieve your information 5 years from now?

Software is not physical, it does not degrade. The key point is recovery of the evidence. The plan must include how to retrieve obsolescence. If a company goes bankrupt, the data is lost. Obsolete file formats? Those should be fixed.

A manufacturer only had Executable Object Code on a box on an aircraft. They had to reverse engineer everything. Not a trivial issue and lots of times this is not handled.

SCM Process *Objectives* (1 of 3)

- "Each configuration item and its successive versions are labeled unambiguously so that a basis is established for the control and reference of configuration items"
- "Baselines are defined for further software life cycle process activity and allow reference to, control of, and traceability between, configuration items"
- "The problem reporting process records process non-compliance with software plans and standards, records deficiencies of outputs of software life cycle processes, records anomalous behavior of software products, and ensures resolution of these problems"

SCM Process *Objectives* (2 of 3)

- "Change control provides for recording, evaluation, resolution, and approval of changes throughout the software life cycle"
- "Change review ensures problems and changes are assessed, approved, or disapproved, approved changes are implemented, and feedback is provided to affected processes through problem reporting and change control methods defined during the software planning process"
- "Status accounting provides data for the configuration management of software life cycle processes with respect to configuration identification, baselines, Problem Reports, and change control"

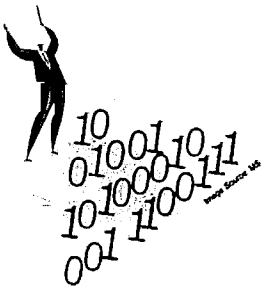
There is no order implied in this list.

SCM Process *Objectives* (3 of 3)

- "Archival and retrieval ensures that the software life cycle data associated with the software product can be retrieved in case of a need to duplicate, regenerate, retest or modify the software product. The objective of the release activity is to ensure that only authorized software is used, especially for software manufacturing, in addition to being archived and retrievable"
- "Software load control ensures that the Executable Object Code and Parameter Data Item Files, if any, are loaded into the *system* or equipment with appropriate safeguards"
- "Software life cycle environment control ensures that the tools used to produce the software are identified, controlled, and retrievable"

Data Control Categories

- Software life cycle data can be assigned to one of two configuration management control categories
  - Control Category 1 (CC1)
  - Control Category 2 (CC2)
- The Annex A tables specify the control category by software level for the software life cycle data items



Section 7.3

8-13

These are the 13 items that vary between C1 and C2 (see next slide):

1. Configuration Identification
2. Baselines
3. Traceability
4. Problem Reporting
5. Change Control - integrity and identification
6. Change Control - tracking
7. Change Review
8. Configuration Status Accounting
9. Retrieval
10. Protection against Unauthorized Changes
11. Media Selection, Refreshing, Duplication
12. Release
13. Data Retention

CC2 is a subset of CC1.

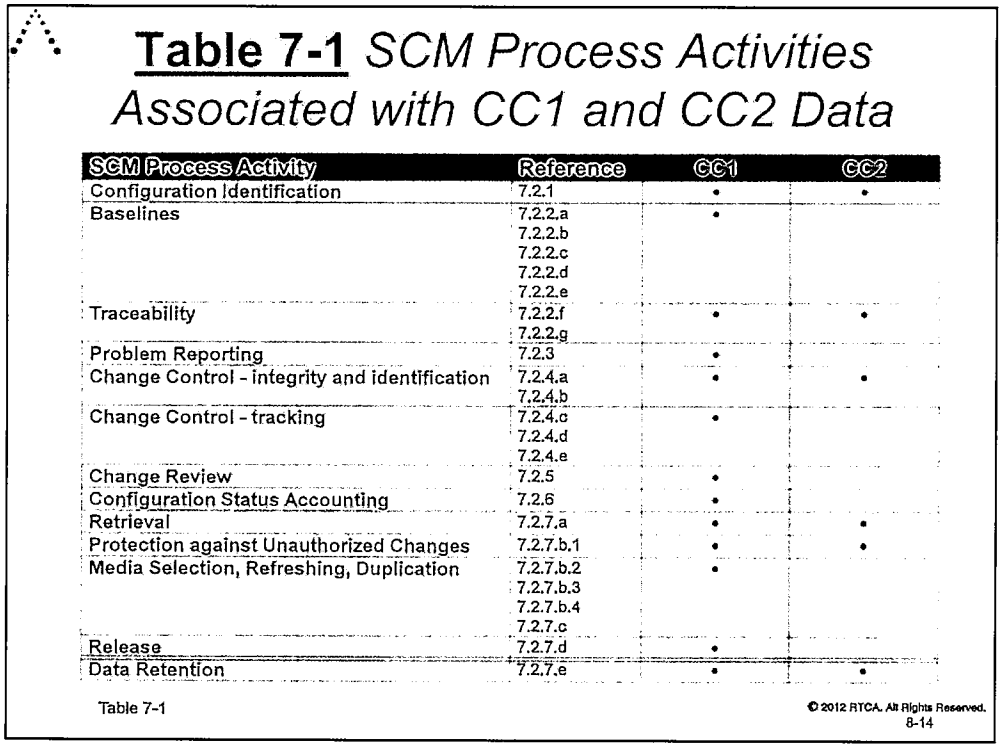


Table 7-1 *SCM Process Activities Associated with CC1 and CC2 Data*

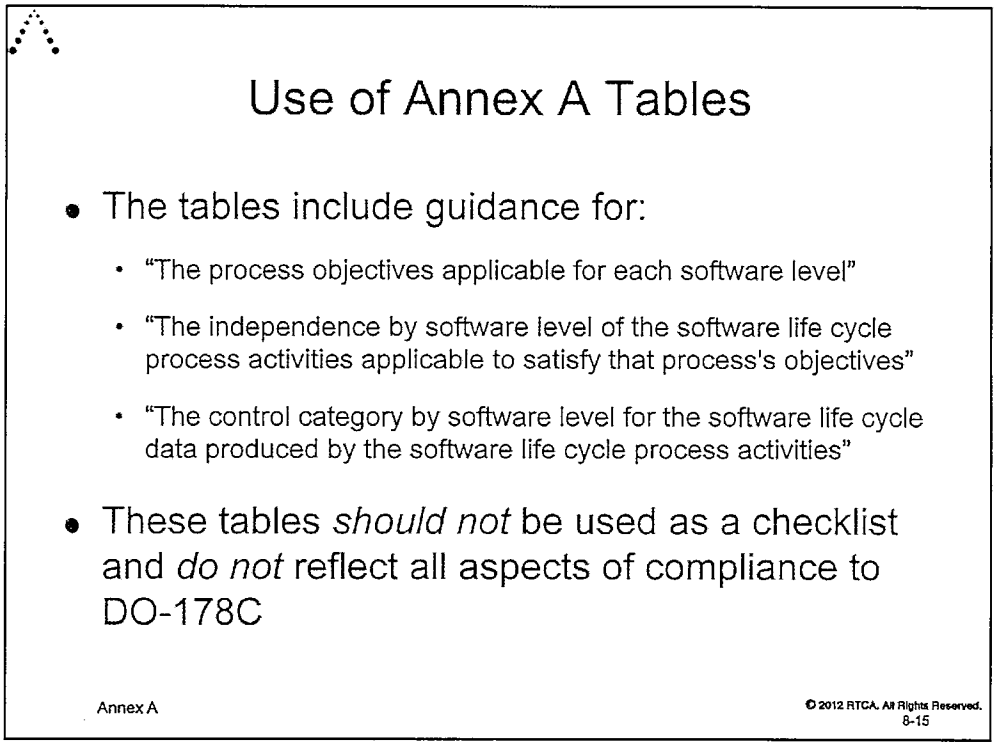
SCM Process Activity	Reference	CC1	CC2
Configuration Identification	7.2.1	•	•
Baselines	7.2.2.a 7.2.2.b 7.2.2.c 7.2.2.d 7.2.2.e	•	
Traceability	7.2.2.f 7.2.2.g	•	•
Problem Reporting	7.2.3	•	
Change Control - integrity and identification	7.2.4.a 7.2.4.b	•	•
Change Control - tracking	7.2.4.c 7.2.4.d 7.2.4.e	•	
Change Review	7.2.5	•	
Configuration Status Accounting	7.2.6	•	
Retrieval	7.2.7.a	•	•
Protection against Unauthorized Changes	7.2.7.b.1	•	•
Media Selection, Refreshing, Duplication	7.2.7.b.2 7.2.7.b.3 7.2.7.b.4 7.2.7.c	•	
Release	7.2.7.d	•	
Data Retention	7.2.7.e	•	•

Table 7-1

8-14

Table 7-1 defines the set of SCM process activities associated with each control category, where • indicates the minimum activities that apply for software life cycle data of that category. CC2 activities are a subset of the CC1 activities.

Even for Level A software, some data items are CC2. For example, you would not want to create a problem report to close a problem report.



In order to fully understand the guidance, the full body of DO-178C should be considered.

A Sample Table (Table A-6)

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
3	Executable Object Code complies with low-level requirements.	6.4.c	6.4.2 6.4.2.1 6.4.3 6.5	●	●	○		Software Verification Cases and Procedures	11.13	①	①	②	
								Software Verification Results	11.14	②	②	②	
								Trace Data	11.21	①	①	②	

LEGEND:	
●	The objective should be satisfied with independence.
○	The objective should be satisfied.
Blank	Satisfaction of objective is at applicant's discretion.
①	Data satisfies the objectives of Control Category 1 (CC1).
②	Data satisfies the objectives of Control Category 2 (CC2).

Annex A

8-16

Most objectives don't vary as much as this example.

The legend applies to "Applicability by Software Level" and "Control Category by Software Level" for all tables.

In this sample table, we can see that the objective must be met with independence for Level A and B software, and is not even *required* for Level D software.

Likewise, the first and third outputs are CC1 control category data items for Level A and B software and, again, are not even *required* for Level D software.

SQA has to be accomplished with organizational independence.

If you are in Level D, which allows for COTS, you don't have to do the data items.

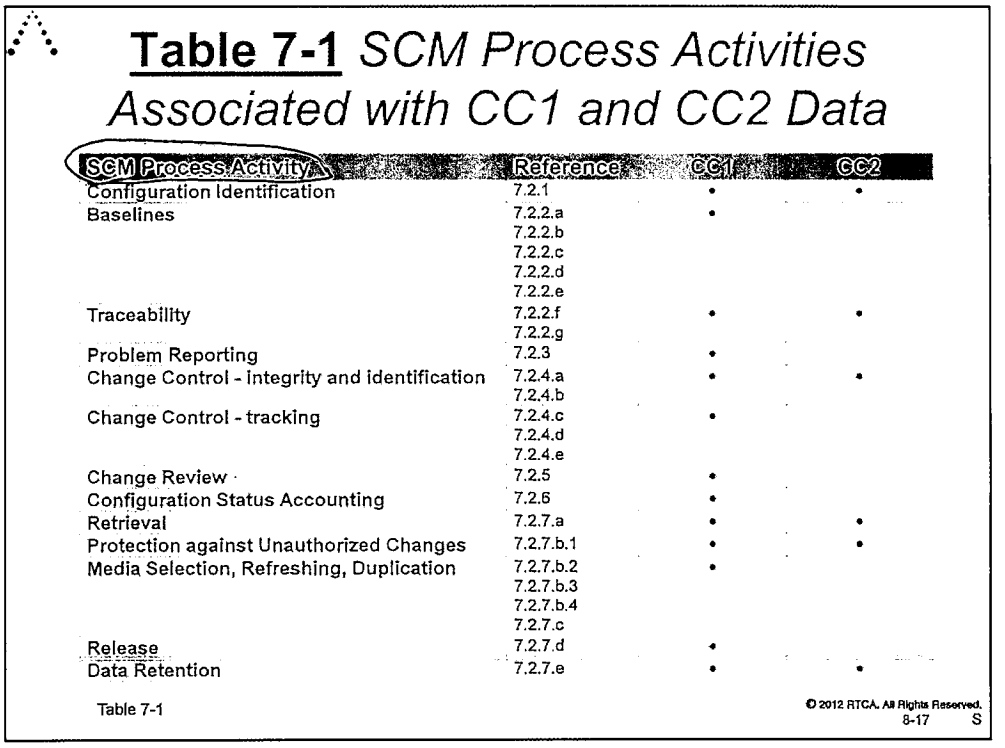


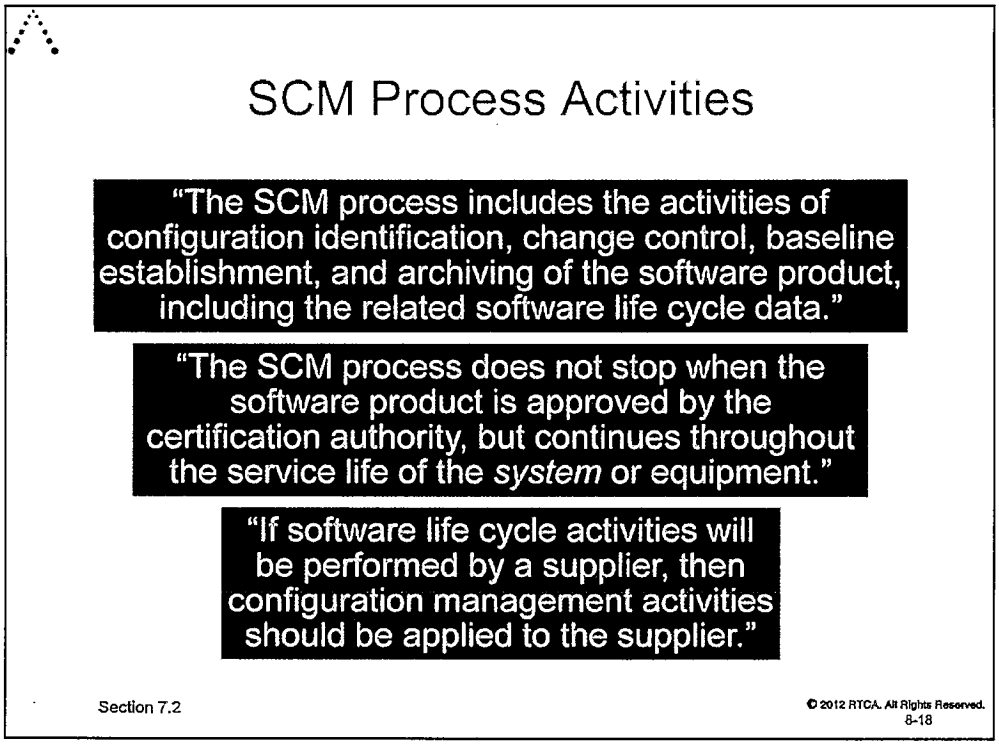
Table 7-1 *SCM Process Activities Associated with CC1 and CC2 Data*

SCM Process Activity	Reference	CC1	CC2
Configuration Identification	7.2.1	•	•
Baselines	7.2.2.a	•	
	7.2.2.b		
	7.2.2.c		
	7.2.2.d		
	7.2.2.e		
Traceability	7.2.2.f	•	•
	7.2.2.g		
Problem Reporting	7.2.3	•	
Change Control - integrity and identification	7.2.4.a	•	•
	7.2.4.b		
Change Control - tracking	7.2.4.c	•	
	7.2.4.d		
	7.2.4.e		
Change Review	7.2.5	•	
Configuration Status Accounting	7.2.6	•	
Retrieval	7.2.7.a	•	•
Protection against Unauthorized Changes	7.2.7.b.1	•	•
Media Selection, Refreshing, Duplication	7.2.7.b.2	•	
	7.2.7.b.3		
	7.2.7.b.4		
	7.2.7.c		
Release	7.2.7.d	•	
Data Retention	7.2.7.e	•	•

Table 7-1

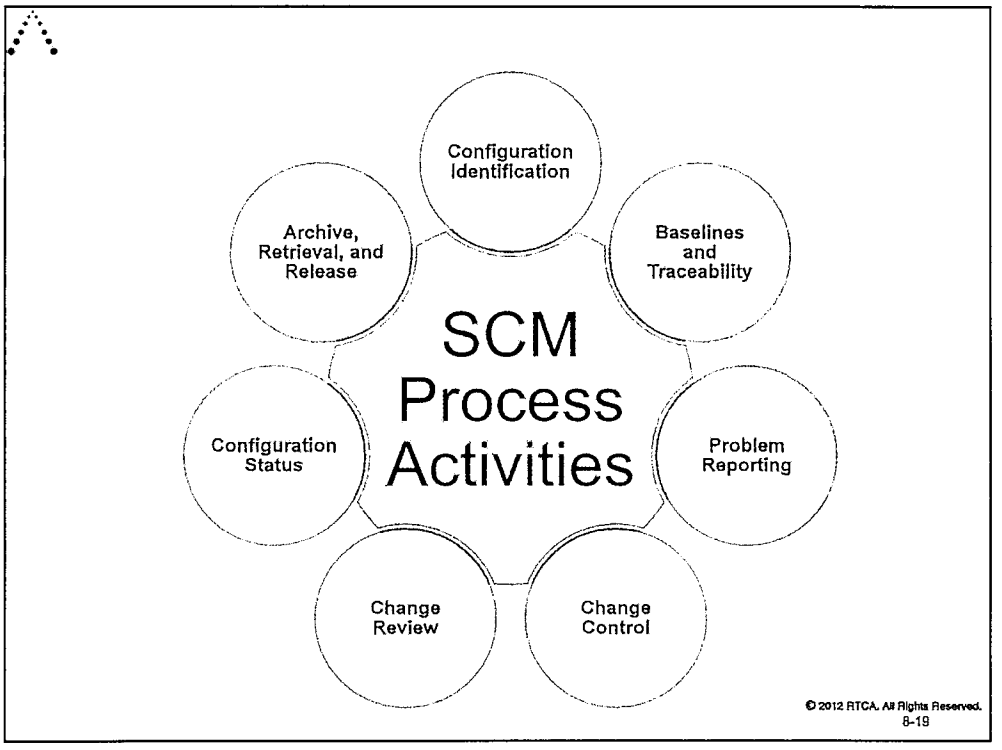
Table 7-1 defines the set of SCM process activities associated with each control category, where • indicates the minimum activities that apply for software life cycle data of that category. CC2 activities are a subset of the CC1 activities.

There are 13 different activities. You don't always have to do all of them. CC2 is a subset of CC1. When you look under Annex A tables for Problem Reports it cascades.

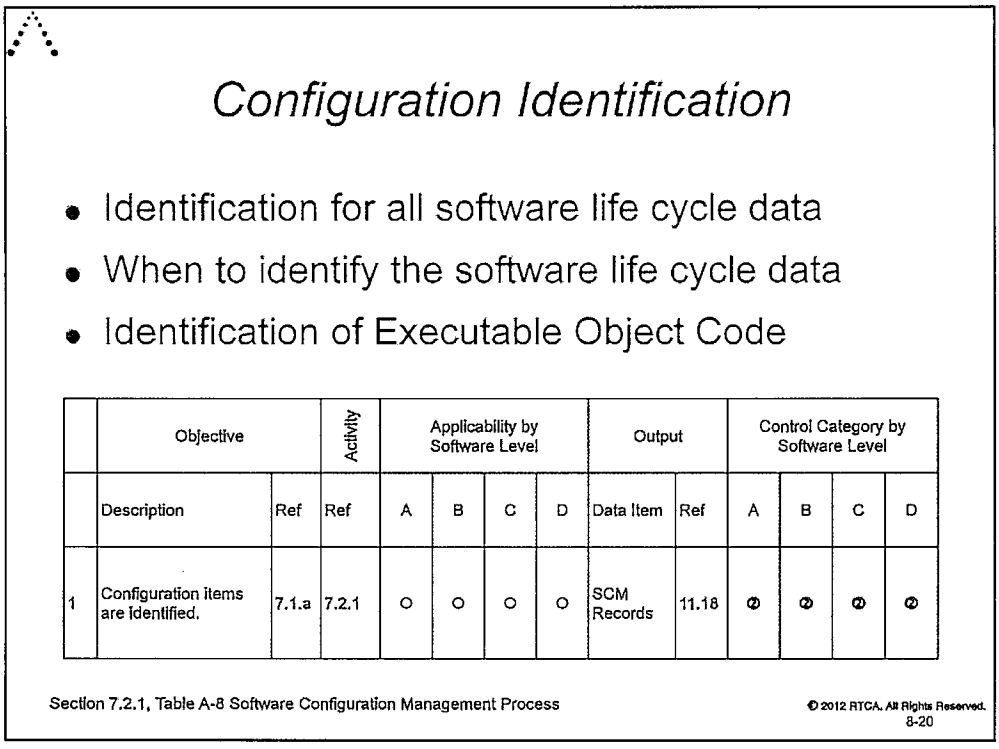


This chart describes the SCM process activities.

- configuration identification,
- change control,
- baseline establishment, and
- archiving of the software product, including the related software life cycle data



These are the 7 SCM Sub-Processes



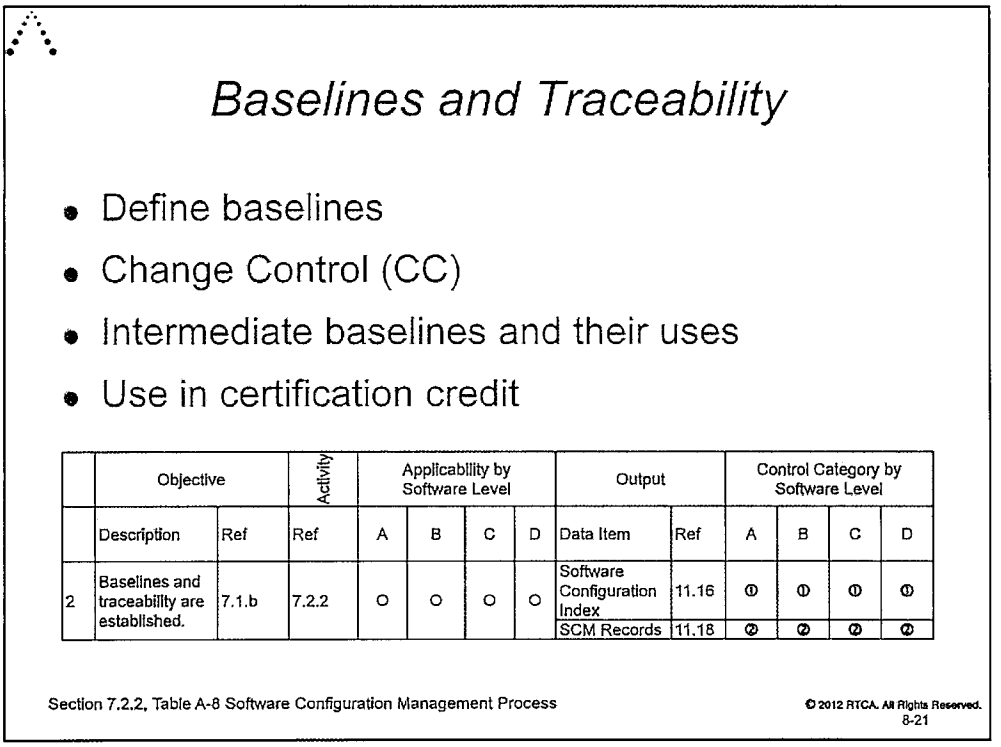
Configuration identification should be established for the software life cycle data.

Configuration identification should be established for each configuration item, for each separately controlled component of a configuration item, and for combinations of configuration items that comprise a software product.

Configuration items should be configuration identified prior to the implementation of change control and traceability analysis.

A configuration item should be configuration identified before that item is used by other software life cycle processes, referenced by other software life cycle data, or used for software manufacture or software loading.

If the software product identification cannot be determined by physical examination (for example, part number plate examination), then the Executable Object Code and Parameter Data Item Files, if any, should contain configuration identification that can be accessed by other parts of the *system* or equipment. This may be applicable to field-loadable software.



	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
2	Baselines and traceability are established.	7.1.b	7.2.2	O	O	O	O	Software Configuration Index	11.16	①	①	①	①
								SCM Records	11.18	②	②	②	②

Baselines should be established for configuration items used for certification credit. Intermediate baselines may be established to aid in controlling software life cycle process activities.

A software product baseline should be established for the software product and defined in the Software Configuration Index.

Note: *User-modifiable software is not included in the software product baseline, except for its associated protection and boundary components. Therefore, modifications may be made to user-modifiable software without affecting the configuration identification of the software product baseline.*

Baselines should be established in controlled software libraries, whether physical, electronic, or other, to ensure their integrity. Once a baseline is established, it should be protected from change.

Change control activities should be followed to develop a derivative baseline from an established baseline.

A baseline should be traceable to the baseline from which it was derived, if certification credit is sought for software life cycle process activities or data associated with the development of the previous baseline.

A configuration item should be traceable to the configuration item from which it was derived, if certification credit is sought for software life cycle process activities or data associated with the development of the previous configuration item.

A baseline or configuration item should be traceable either to the output it identifies or to the process with which it is associated.

Problem Reporting, Tracking and Corrective Action

- Problem(s) found during verification and by Software Quality Assurance (SQA) must be documented
- Problem reports must identify item(s) in error
- Corrective action(s) must be documented

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
3	Problem reporting, change control, change review, and configuration status accounting are established.	7.1.c 7.1.d 7.1.e 7.1.f	7.2.3 7.2.4 7.2.5 7.2.6	O	O	O	O	Problem Reports	11.17	②	②	②	②
								SCM Records	11.18	②	②	②	②

Section 7.2.3, Table A-8 Software Configuration Management Process

8-22

A Problem Report should be prepared that describes the process non-compliance with plans, output deficiency, or software anomalous behavior, and the corrective action taken.

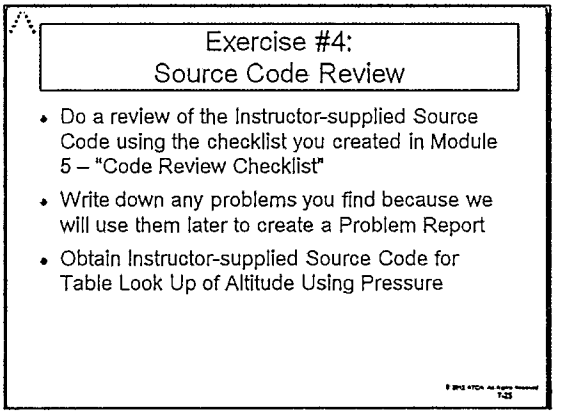
Note: *Software life cycle process and software product problems may be recorded in separate problem reporting systems.*

Problem reporting should provide for configuration identification of affected configuration item(s) or definition of affected process activities, status reporting of Problem Reports, and approval and closure of Problem Reports.

Problem Reports that require corrective action of the software product or outputs of software life cycle processes should invoke the change control activity

Exercise #6: **Problem Report**

- Create a problem report from the results of the Module 7 Source Code Review exercise.



8-23

Instructor will ask the class what should be included in the Problem Report
Instructor will ask the class for a problem they found in the code review exercise.
The problem found should be turned into a formal report.

Will be done on a White Board or Flip Chart.

PRid:
Date:
Author:
Verification Results id:
Description of problem:
Description of solution.
Steps taken to implement & verify solution.

Change Control

- Baselines and configuration items must be protected from change (versioning)
- Changes to a configuration item (new version) causes the software life cycle processes to be repeated from the origin of the error
- All changes to baselines and configuration items

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
3	Problem reporting, change control, change review, and configuration status accounting are established.	7.1.c 7.1.d 7.1.e 7.1.f	7.2.3 7.2.4 7.2.5 7.2.6	O	O	O	O	Problem Reports SCM Records	11.17 11.18	②	②	②	②

Section 7.2.4

8-24

Change control should preserve the integrity of the configuration items and baselines by providing protection against their change.

Change control should ensure that any change to a configuration item requires a change to its configuration identification.

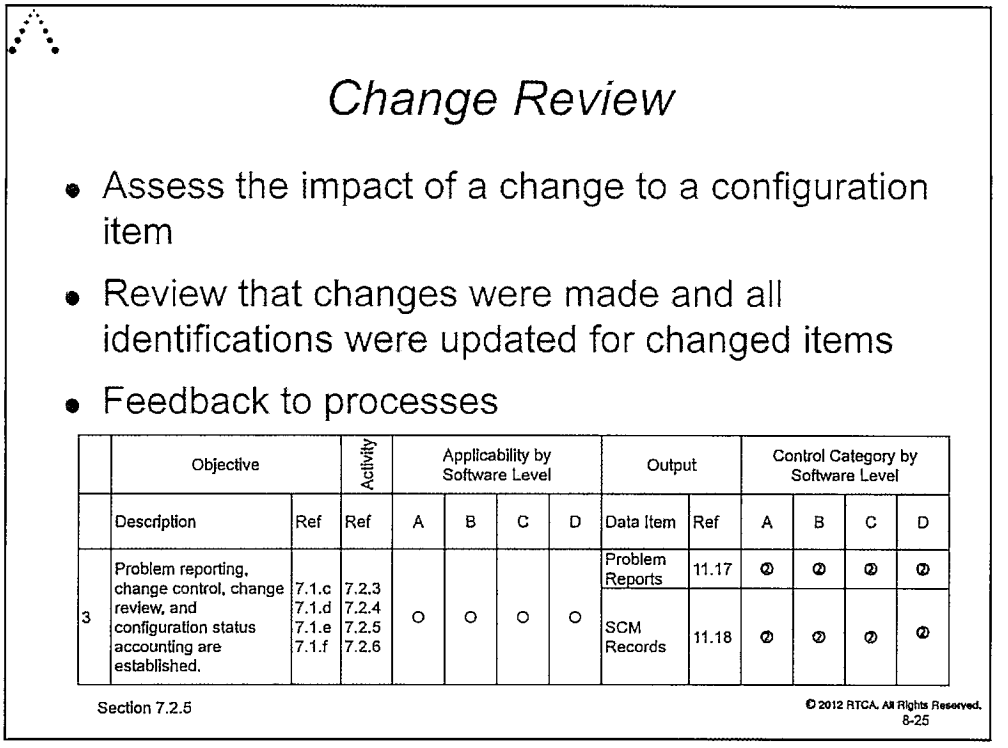
Changes to baselines and to configuration items under change control to produce derivative baselines should be recorded, approved, and tracked. Problem reporting is related to change control, since resolution of a reported problem may result in changes to configuration items or baselines.

Note: *It is generally recognized that early implementation of change control assists the control and management of software life cycle process activities.*

Software changes should be traced to their origin and the software life cycle processes repeated from the point at which the change affects their outputs. For example, an error discovered at hardware/software integration, that is shown to result from an incorrect design, should result in design correction, code correction, and repetition of the associated integral process activities.

Throughout the change activity, software life cycle data affected by the change should be updated and records should be maintained for the change control activity.

The change control activity is aided by the change review activity.



Change Review

- Assess the impact of a change to a configuration item
- Review that changes were made and all identifications were updated for changed items
- Feedback to processes

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
3	Problem reporting, change control, change review, and configuration status accounting are established.	7.1.c 7.1.d 7.1.e 7.1.f	7.2.3 7.2.4 7.2.5 7.2.6	O	O	O	O	Problem Reports	11.17	②	②	②	②
								SCM Records	11.18	②	②	②	②

Section 7.2.5

8-25

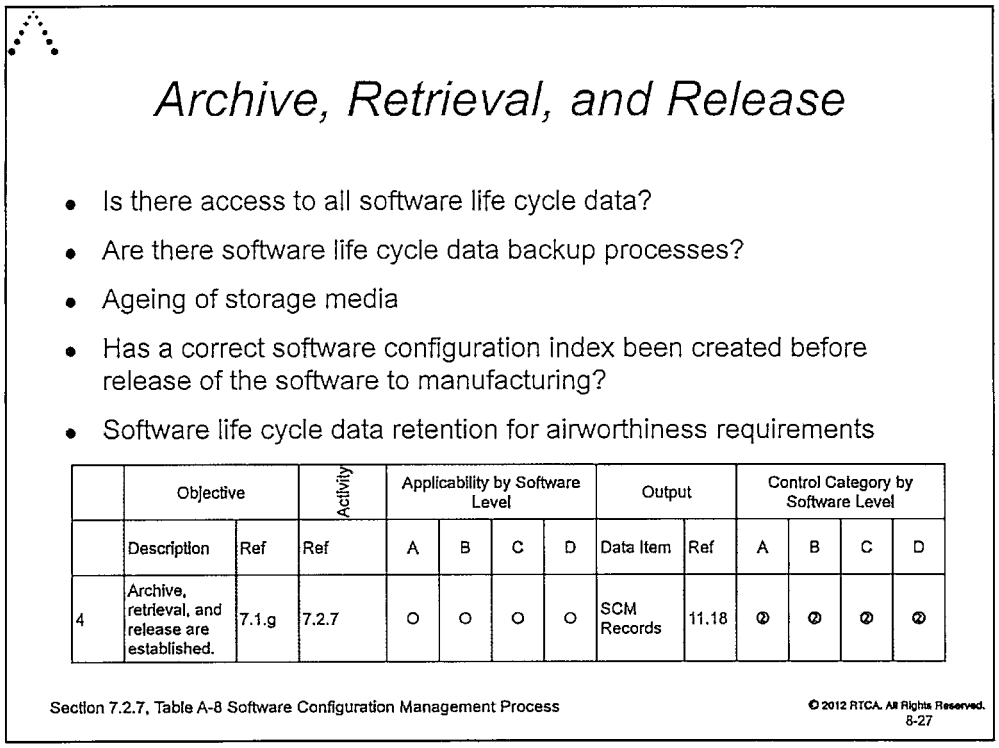
- Assessment of the impact of the problem or proposed change on *system* requirements. Feedback should be provided to the *system* processes, including the *system* safety assessment process, and any responses from the *system* processes should be assessed.
- Assessment of the impact of the problem or proposed change on software life cycle data identifying changes to be made and actions to be taken.
- Confirmation that affected configuration items are configuration identified.
- Feedback of Problem Report or change impact and decisions to affected processes.

Configuration Status Accounting

- "Reporting on configuration item identification, baseline identification, Problem Report status, change history, and release status."
- "Definition of the data to be maintained and the means of recording and reporting status of this data."

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
3	Problem reporting, change control, change review, and configuration status accounting are established.	7.1.c 7.1.d 7.1.e 7.1.f	7.2.3 7.2.4 7.2.5 7.2.6	O	O	O	O	Problem Reports	11.17	②	②	②	②
								SCM Records	11.18	②	②	②	②

Section 7.2.6, Table A-8 Software Configuration Management Process



Software life cycle data associated with the software product should be retrievable from the approved source (for example, an archive at the developing organization or company).

Procedures should be established to ensure the integrity of the stored data, regardless of medium of storage, by:

- Ensuring that no unauthorized changes can be made.
- Selecting storage media that minimize regeneration errors or deterioration.
- Preventing loss or corruption of data over time. Depending on the storage media used, this may include periodically exercising the media or refreshing the archived data.
- Storing duplicate copies in physically separate archives that minimize the risk of loss in the event of a disaster.
- The duplication process should be verified to produce accurate copies, and procedures should exist that ensure error-free copying of the Executable Object Code and Parameter Data Item Files, if any

Configuration items should be identified and released prior to use for software manufacture and the authority for their release should be established. As a minimum, the components of the software product loaded into the airborne *system* or equipment should be released. This includes the Executable Object Code and Parameter Data Item Files, if any, and may also include associated media for software loading.

Note: *Release is generally also required for the data that defines the approved software for loading into the airborne system or equipment. Definition of that data is outside the scope of this document, but may include the Software Configuration Index.*

Data retention procedures should be established to satisfy airworthiness requirements and enable software modifications.

Note: *Additional data retention considerations may include items such as business needs and future certification authority reviews, which are outside the scope of this document.*

Software Load Control

- Part numbering
- Compatibility with airborne *system* (What aircraft can use this software?)

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
5	Software load control is established.	7.1.h	7.4	O	O	O	O	SCM Records	11.18	②	②	②	②

Section 7.4, Ref. Table A-8 Software Configuration Management Process

8-28

Software load control refers to the process by which programmed instructions and data are transferred from a master memory device into the *system* or equipment. For example, methods may include (subject to approval by the certification authority) the installation of factory pre-programmed memory devices or "in situ" re-programming of the *system* or equipment using a field loading device. Whichever method is used, software load control should include:

- Procedures for part numbering and media identification that identify software configurations that are intended to be approved for loading into the airborne *system* or equipment.
- Whether the software is delivered as an end item or is delivered installed in the airborne *system* or equipment, records should be kept that confirm software compatibility with the airborne *system* or equipment hardware.

Software Life Cycle Environment Control

- Identify all tools used in the software life cycle processes
- Controlling qualified tools
- Controlling non-qualified tools

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
6	Software life cycle environment control is established.	7.1.i	7.5	O	O	O	O	Software Life Cycle Environment Configuration Index	11.15	①	①	①	②
								SCM Records	11.18	②	②	②	②

Section 7.5, Ref. Table A-8 Software Configuration Management Process

8-29

Configuration identification should be established for the Executable Object Code, or equivalent, of the tools used to develop, control, build, verify, and load the software.

The SCM process for controlling qualified tools should comply with the objectives associated with Control Category 1 or Control Category 2 data.

Unless section 7.5b applies, the SCM process for controlling the Executable Object Code, or equivalent, for tools used to build and load the software (for example, compilers, assemblers, and linkage editors) should comply with the objectives associated with Control Category 2 data, as a minimum.

Review Questions

- When should you start configuration management of requirements?
  - A. Immediately after the configuration management plan is approved
  - B. When you start testing
  - C. When your problem reporting system is ready
  - D. When you transition from the requirements phase to the coding phase
  - E. Immediately upon creation of the first High-level requirement
  - F. Whenever your plan requires you to

Review Questions

Explain the chicken and egg problem with the Software Configuration Index:

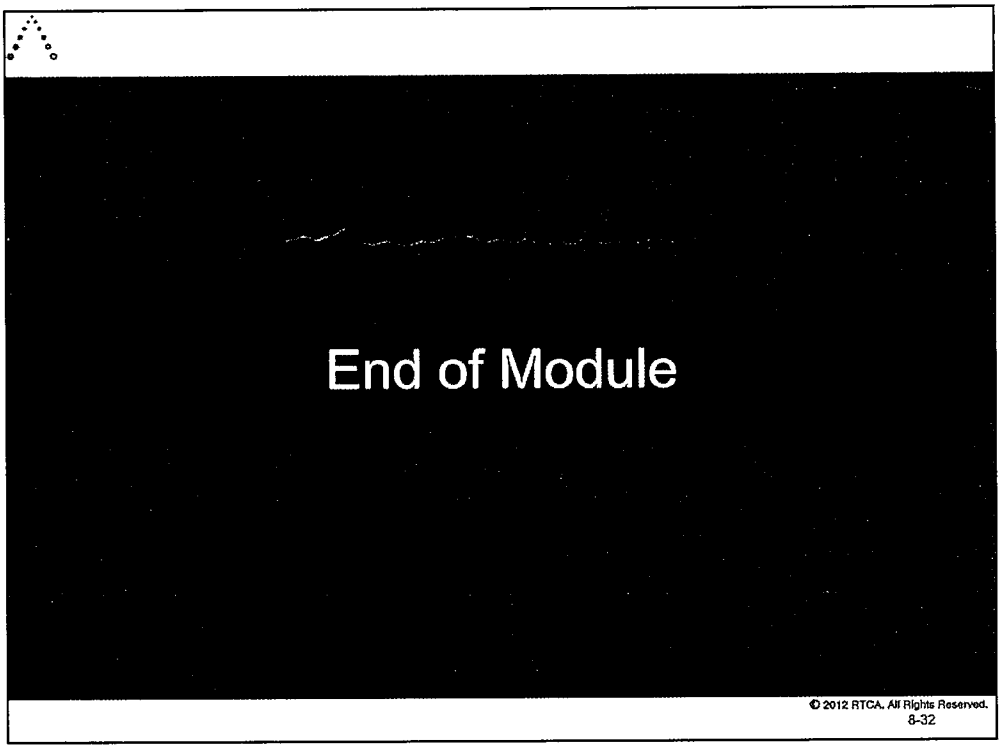
- The Software Configuration Index must include the Software Accomplishment Summary
- The Software Accomplishment Summary must reference the Software Configuration Index
- How do you catalogue the catalogue?

8-31

Answer

The Software Configuration Index must identify all Software Life Cycle Data that were created from the Software Life Cycle processes. This includes the Software Accomplishment Summary. The Software Accomplishment Summary must explicitly reference "by configuration identifiers and version, the applicable Software Configuration Index".

So to avoid the problem of changes to one causing changes to the other which causes changes to the first and so on, both must be updated with the future identifier and version of the other. This may not be as easy as it seems and a check that the two documents agree must be done after both are released.


End of Module
© 2012 RTCA, All Rights Reserved.
8-32

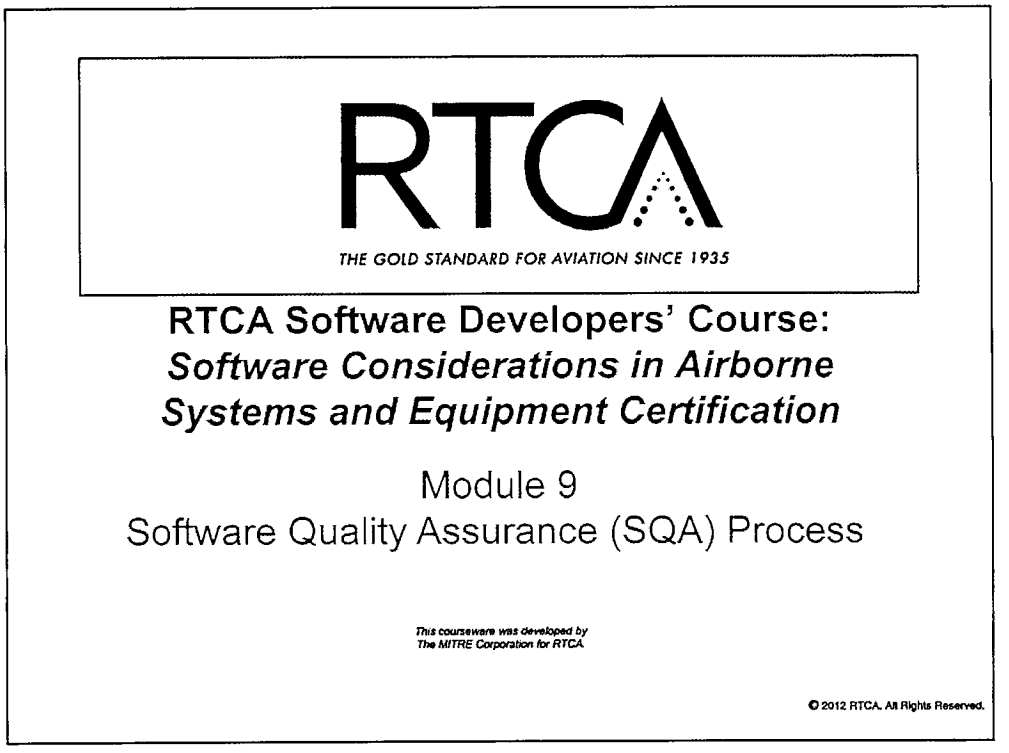
Exercise # 6
Problem Report

Create a problem report from the results of the Code Review exercise in Module 7.

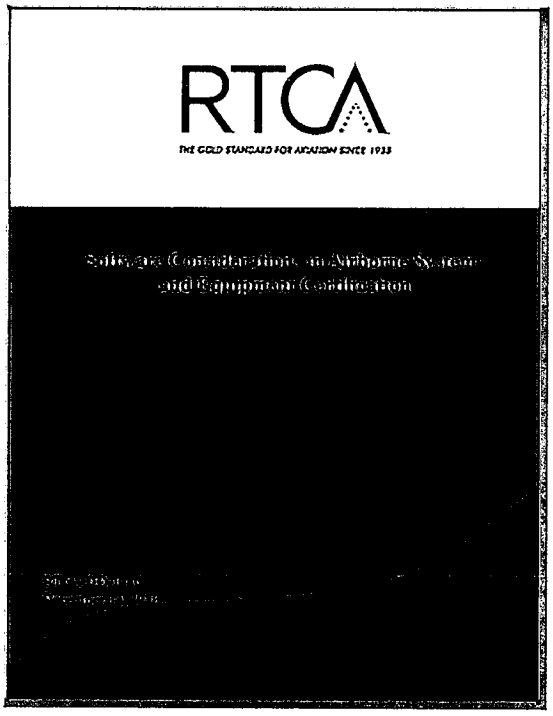
Discuss what needs to be included in the problem report.

Answer Template

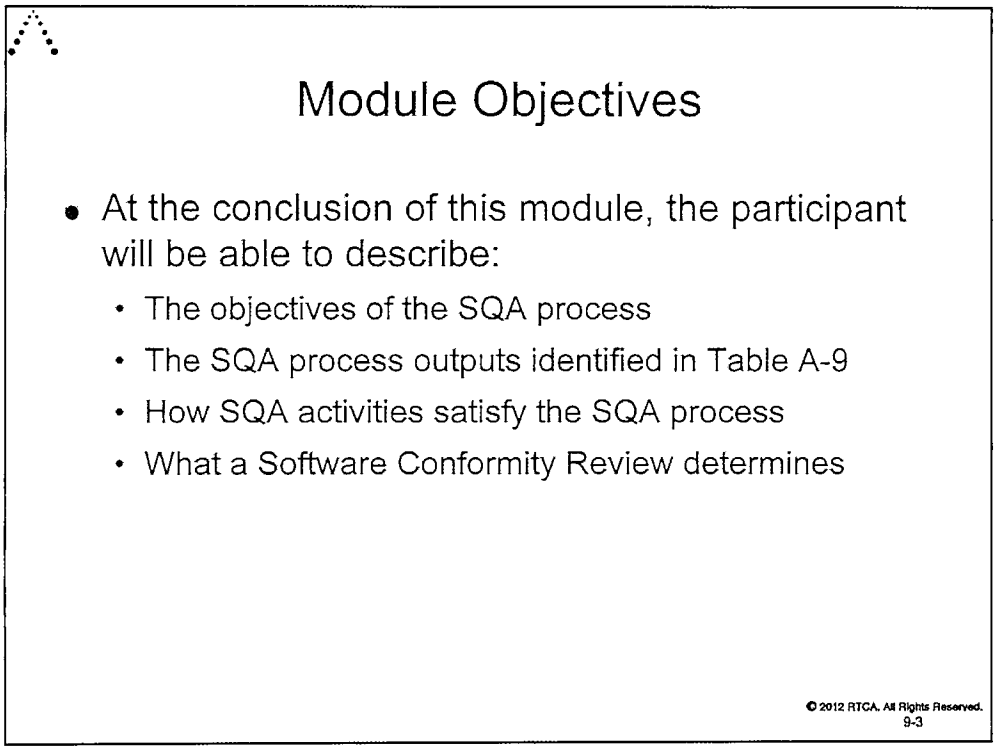
PR id:
Date:
Author:
Verification results id:
Description of problem:
Description of solution:
Steps taken to implement and verify solution:
Approvals for closure:



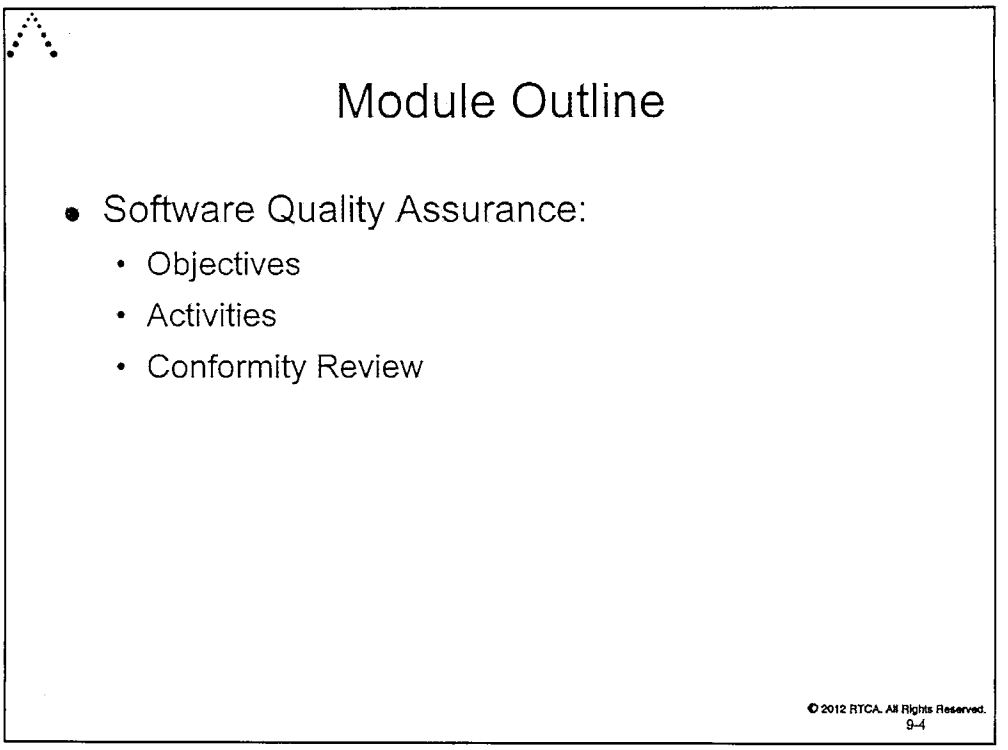
This 1-hour module introduces the Software Quality Assurance Process to provide confidence that the software life cycle processes produce software that conforms to the software requirements by assuring that the these processes are performed in compliance with approved software plans and standards. For this process, objectives, outputs, activities and the Conformity Review are described. The topics covered are the process audits, artifact audits, transition records, deviation records and Conformity Review.



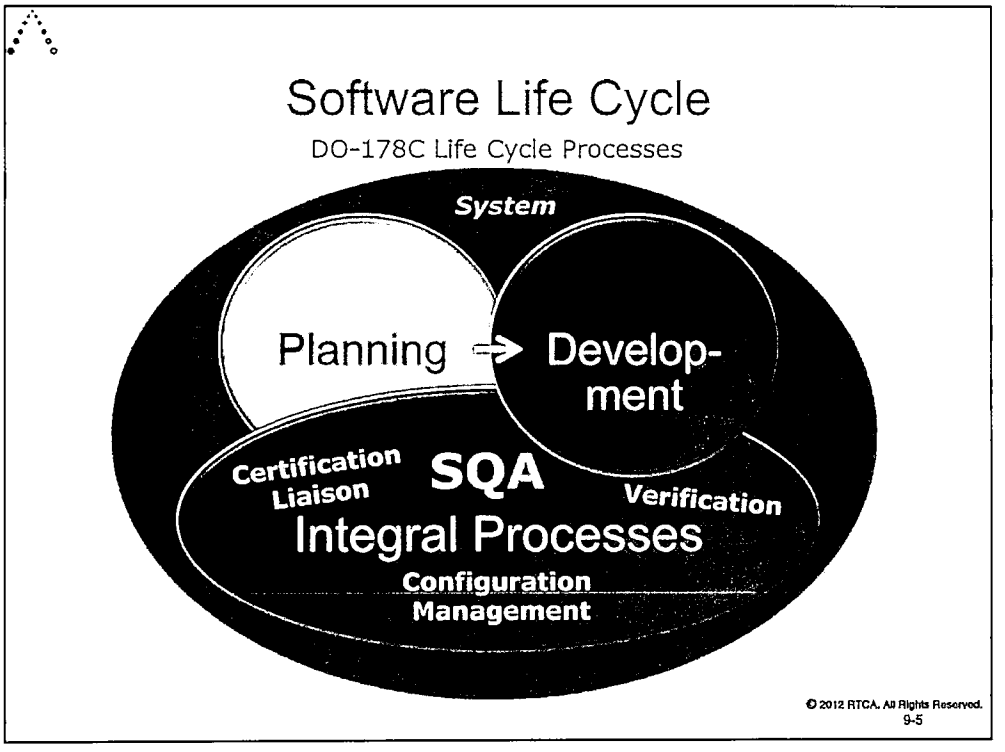
Section 8



This chart provides the objectives of Module 9.

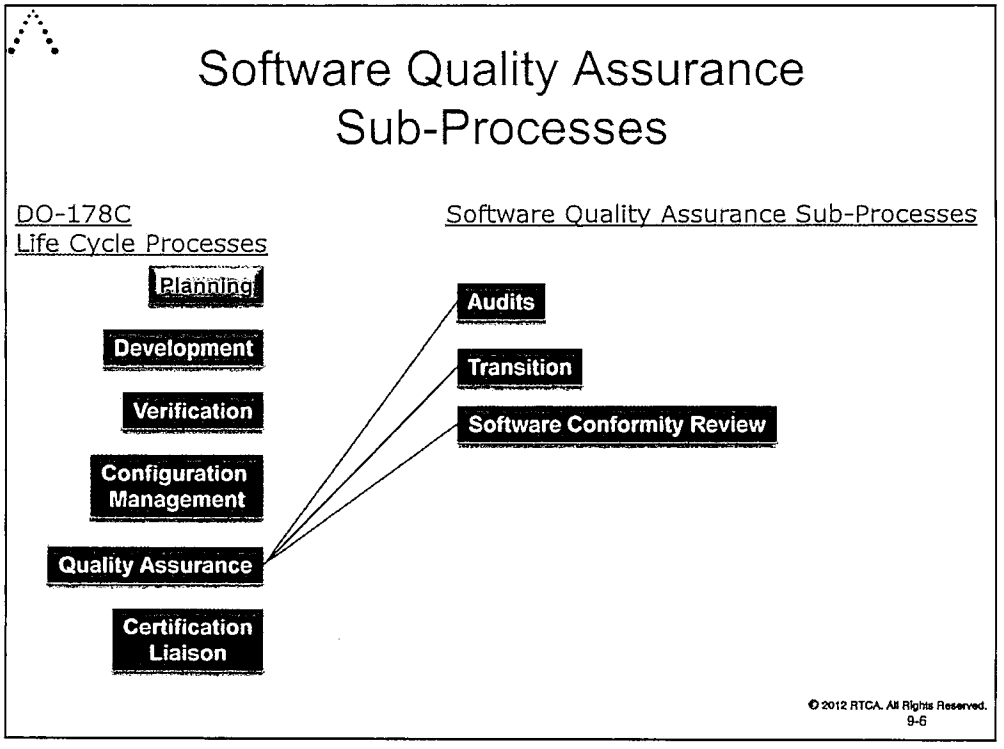


This chart outlines Module 9. This module is focused on Objectives, Activities and Conformity Reviews as they relate to DO-178C.

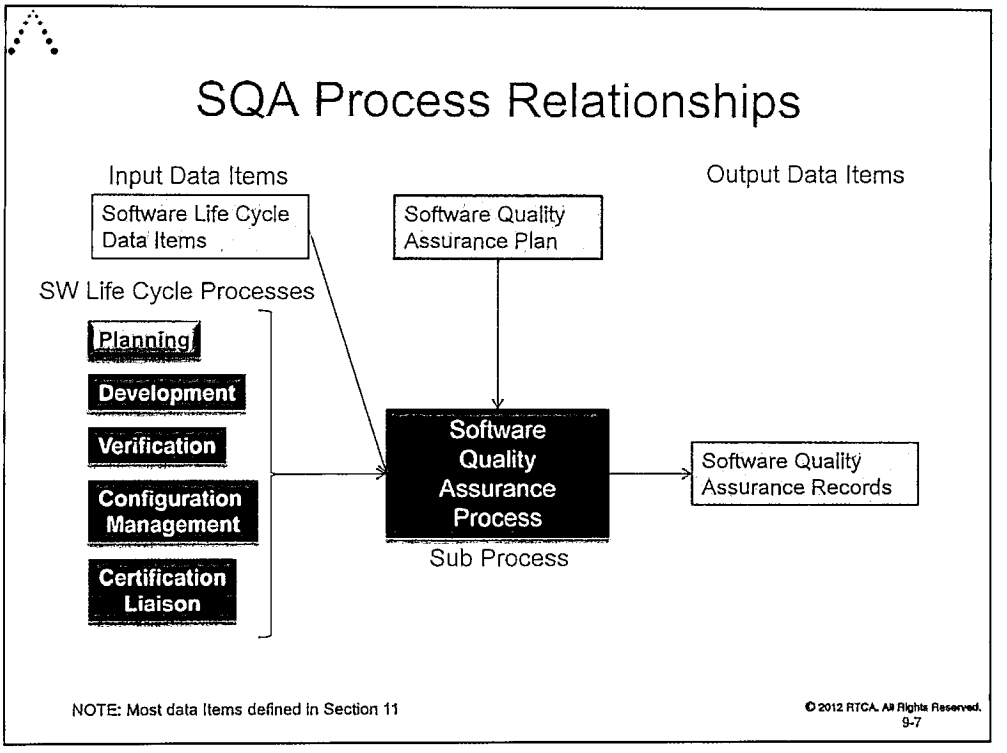


There are many life cycle process charts.

This slide depicts the life cycle processes in DO-178C that are used for software.



Audits, Transition and Software Conformity Review are the three software sub-processes of QA.



This chart describes the SQA process relationships, along with input and output data items, gained in accordance with the SQA plan.

SQA Process

The SQA process assesses the software life cycle processes and their outputs to obtain assurance that:

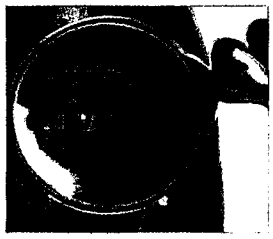


Image Source: MS

- objectives are satisfied
- deficiencies are detected, evaluated, tracked, and resolved
- software product and software life cycle data conform to certification requirements

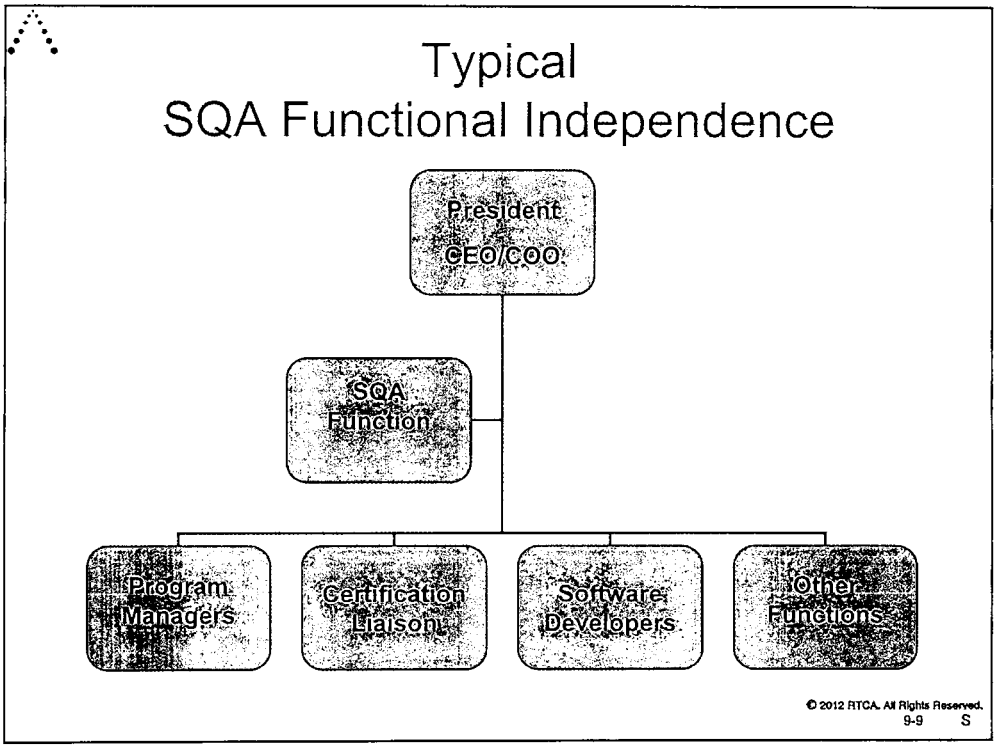
The SQA process is performed with **independence**

Section 8

9-8

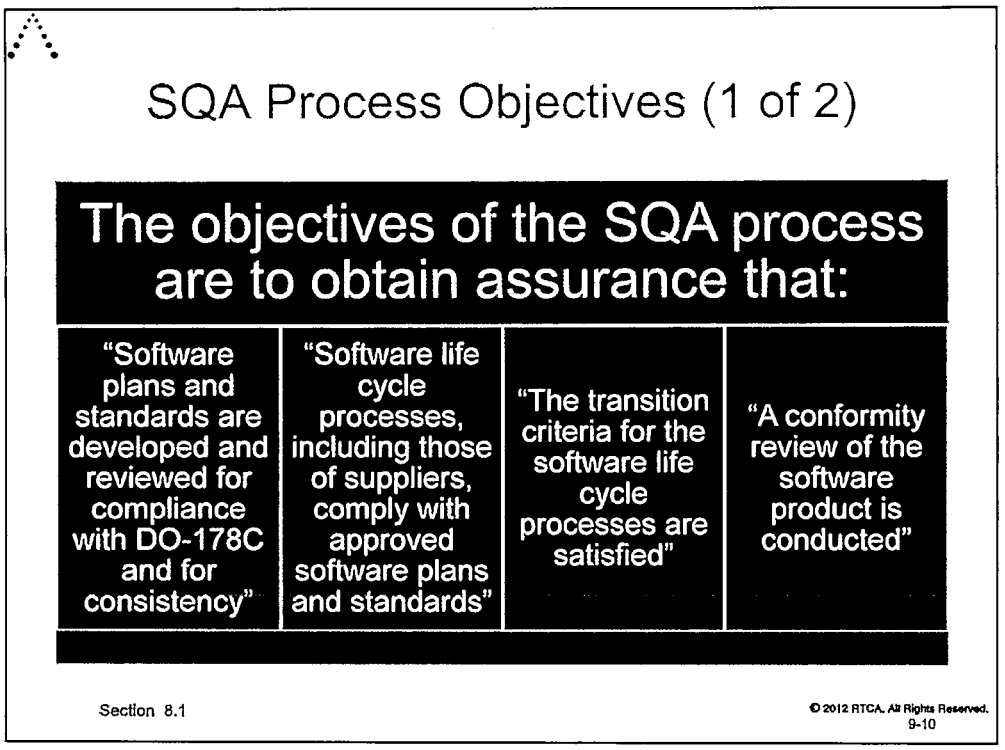
Independence – Separation of responsibilities which ensures the accomplishment of objective evaluation. (1) For software verification process activities, independence is achieved when the verification activity is performed by a person(s) other than the developer of the item being verified, and a tool(s) may be used to achieve equivalence to the human verification activity. (2) For the software quality assurance process, independence also includes the authority to ensure corrective action.

The organization chart shows how SQA can have independence within the organization.

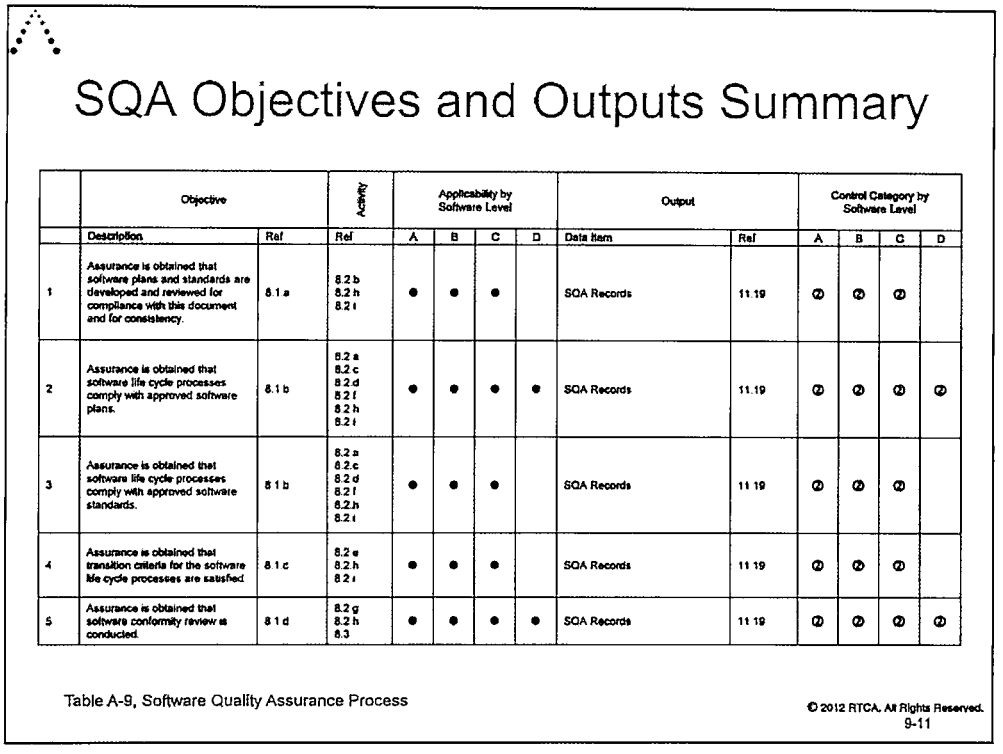


The organization chart shows how SQA can have independence within the organization.

SQA should report to senior management and not to the organizations that it reviews.



The SQA process objectives provide confidence that the software life cycle processes produce software that conforms to its requirements by assuring that these processes are performed in compliance with the approved software plans and standards.

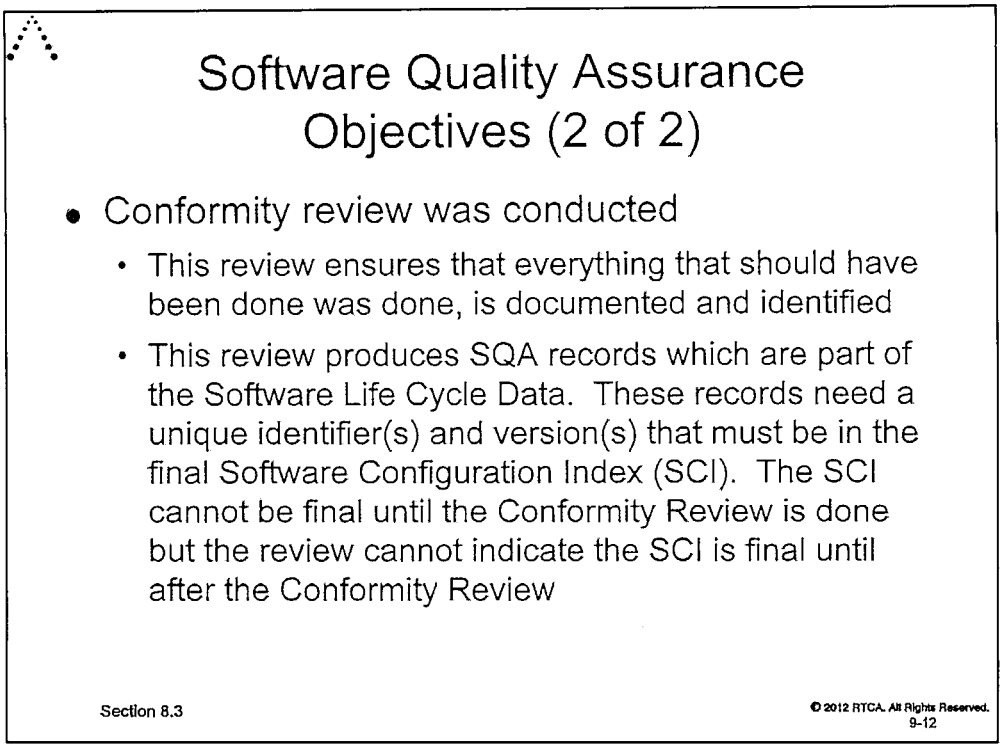


SQA Objectives and Outputs Summary

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	Assurance is obtained that software plans and standards are developed and reviewed for compliance with this document and for consistency.	8.1 a	8.2 b 8.2 h 8.2 i	●	●	●		SQA Records	11.19	②	②	②	
2	Assurance is obtained that software life cycle processes comply with approved software plans.	8.1 b	8.2 a 8.2 c 8.2 d 8.2 f 8.2 h 8.2 i	●	●	●	●	SQA Records	11.19	②	②	②	②
3	Assurance is obtained that software life cycle processes comply with approved software standards.	8.1 b	8.2 a 8.2 c 8.2 d 8.2 f 8.2 h 8.2 i	●	●	●		SQA Records	11.19	②	②	②	
4	Assurance is obtained that transition criteria for the software life cycle processes are satisfied.	8.1 c	8.2 e 8.2 h 8.2 i	●	●	●		SQA Records	11.19	②	②	②	
5	Assurance is obtained that software conformity review is conducted.	8.1 d	8.2 g 8.2 h 8.3	●	●	●	●	SQA Records	11.19	②	②	②	②

Table A-9, Software Quality Assurance Process

Table A-9 of Annex A is a summary of the objectives and outputs of the SQA process.



To solve this problem care must be taken with the release of the SQA Records and the Software Configuration Index. Any identifiers and versions used in either data items must be in agreement.

If it isn't identified, it was not done.

SQA Activities (1 of 5)

RAA. (Responsibility, Authority,

- Active role in software verification processes with the authority, responsibility, and independence to ensure that the SQA process objectives are satisfied

  Examples:

  - Closing problem reports
  - Participating on reviews
  - Reviewing *all* problem reports
  - Witnessing testing

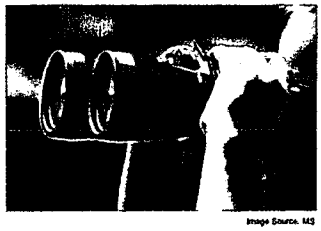


Image Source: IAS

Section 8.2

9-13

Level of activity is up to you.

Responsibility, Accountability, and Authority. This SQA has to have the RAA. It is not a paper tiger organization

All problem reports should become "closed reports", unless determined to have no safety effect by the *system*.

If a Project Lead deviates from the plan, for example, using a different simulator, SQA needs to approve the deviation from the processes.

Suppose a company has short (thin on detail) plans and work orders that describe their processes. They assert that in their Plan, the plans are all that are required, and the meat of their processes were off-site. No, this would not be certified.

SQA Activities (2 of 5)

- Determining that plans and standards exist and are adequate
  - Signing plans and standards

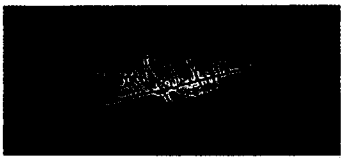


Image Source: M3

- Provide assurance that the software life cycle processes comply with the approved software plans and standards

This gets into evaluation, not just auditing. Are the plans and standards adequate?

SQA Activities (3 of 5)

- Audit software life cycle processes to assure:
  - Software plans are available
  - Detection, recording, evaluating, tracking and resolution of process problems
  - Approved process deviations are recorded
  - Transition criteria were followed
  - Configuration Management processes were followed
  - Any supplier processes and outputs comply with approved plans and standards

Image Source: MS

Section 8.2

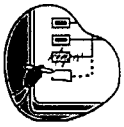
It is generally accepted that early detection of process deviations assists efficient achievement of software life cycle process objectives.

Monitoring of the activities of software life cycle processes may be performed to provide assurance that the activities are under control.

The SQA process should provide assurance that supplier processes and outputs comply with approved software plans and standards.

SQA Activities (4 of 5)

Provide assurance that:



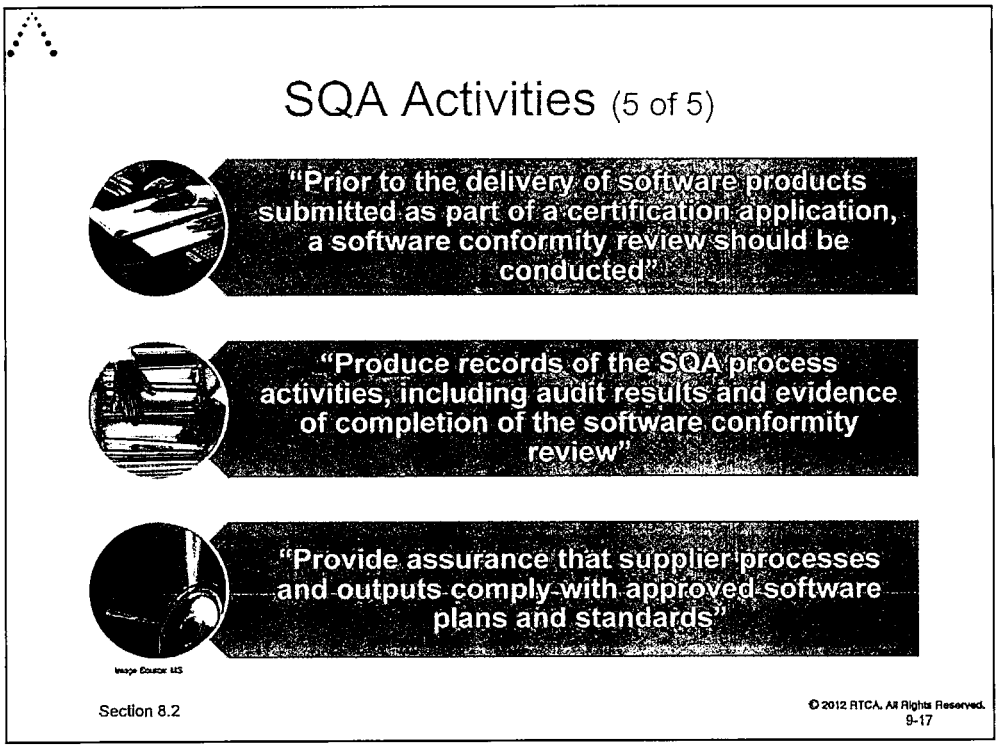
The transition criteria for the software life cycle processes have been satisfied in compliance with the approved software plans

Software life cycle data is controlled in accordance with the control categories and the tables of Annex A

Section 8.2

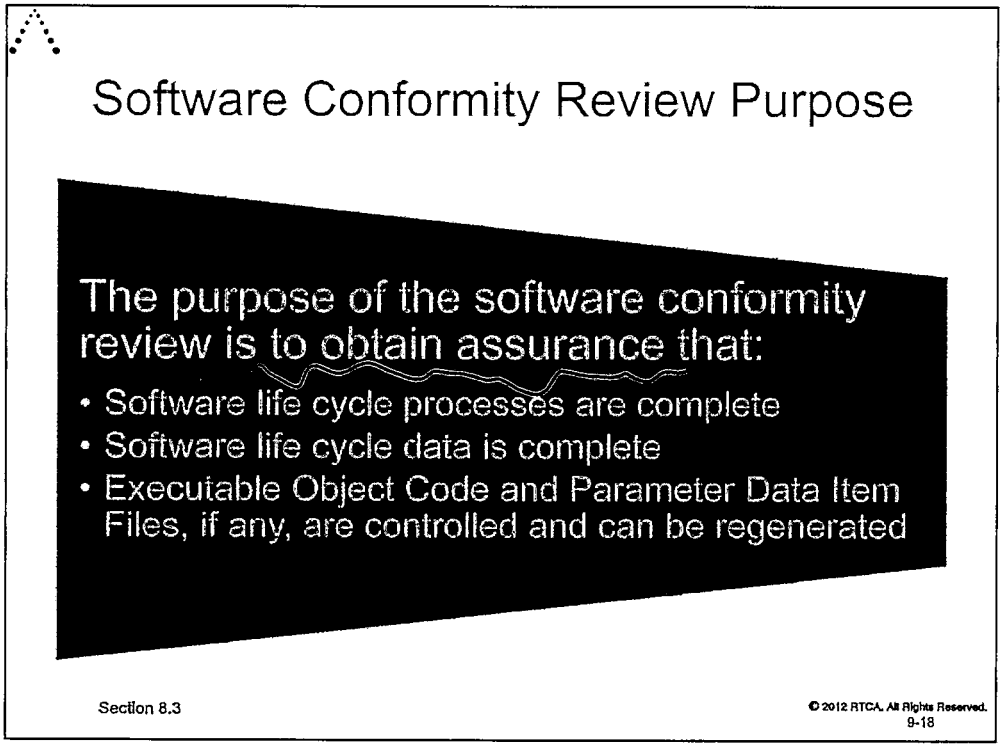
9-16

This determines the actual Transition Criteria were satisfied.



The word must is not used in DO-178C. Must is when you use DO-178C as a means, even though it is not stated. This is a philosophical problem, it is up to company doing the work; they must say the how they will do this review.

Conducting a Conformity Review is a shall. It is not an audit. It means that the processes are complying.



This is to obtain assurance, that life cycle data is complete.

Software Conformity Review (1 of 3)

- *This review should determine that:*
  - All planned software life cycle processes are complete
    - Except for this review
  - All planned software life cycle data was developed and is uniquely identified and controlled

  - Trace data for all software life cycle data is complete and correct

Section 8.3

9-19

Planned software life cycle process activities for certification credit, including the generation of software life cycle data, have been completed and records of their completion are retained.

Software life cycle data developed from specific *system* requirements, safety-related requirements, or software requirements are traceable to those requirements.

Evidence exists that software life cycle data have been produced in accordance with software plans and standards, and is controlled in accordance with the SCM Plan.

Software Conformity Review (2 of 3)

- The Executable Object Code and Parameter Data Item Files, if any, can be regenerated from the archived Source Code
  - This may not be possible with modern compilers
  - All identified configuration items were the items used to generate the Executable Object Code that was verified

- All problem reports, including those from previous baselines, have been evaluated and have their status recorded

Section 8.3

9-20

The Executable Object Code and Parameter Data Item Files, if any, can be regenerated from the archived Source Code.

Evidence exists that Problem Reports have been evaluated and have their status recorded.

Problem Reports deferred from a previous software conformity review are re-evaluated to determine their status.

You have to make sure that the tool can't be changed, otherwise you have to qualify that tool.

Compiler can insert an error without you knowing it. If you don't own the tool, you may not be able to fix this. Take care that you control so you know that what goes out the door is what you tested. The environment the development was done in is important. You may have to deal with it in another way.

If the Sun Work station goes away, what do you do then? Make sure your processes and systems hang together. A date will allow time tracking to prove information happened post another event.

Configuration Management Tool + Development tool 2.

Software Conformity Review (3 of 3)

- All deviations to software life cycle processes are recorded and approved

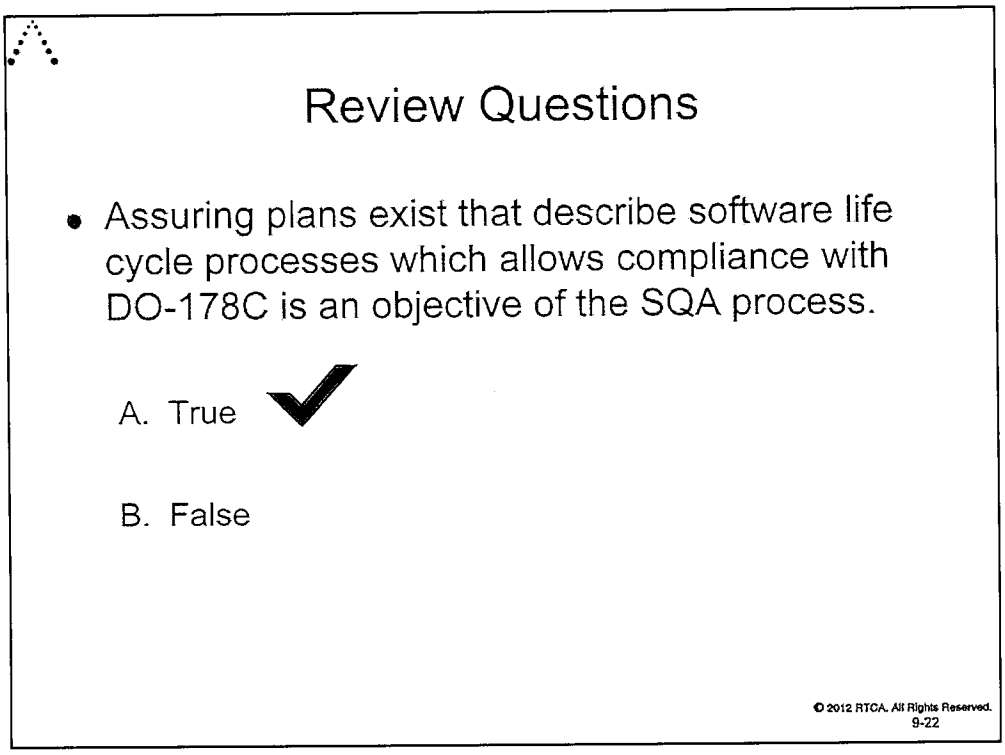
- The release process can load the identified software into the *system*
- If a previous software baseline was used then the current can be traced with approved changes to the previous baseline

Section 8.3

Software requirement deviations are recorded and approved.

The approved software can be loaded successfully through the use of released instructions.

If certification credit is sought for the use of previously developed software, the current software product baseline is traceable to the previous baseline and the approved changes to that baseline.



1 Assurance is obtained that software plans and standards are developed and reviewed for compliance with this document and for consistency.

2 Assurance is obtained that software life cycle processes comply with approved software plans.

3 Assurance is obtained that software life cycle processes comply with approved software standards.

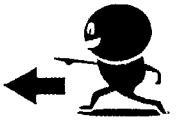
4 Assurance is obtained that transition criteria for the software life cycle processes are satisfied.

5 Assurance is obtained that software conformity review is conducted.

Review Questions

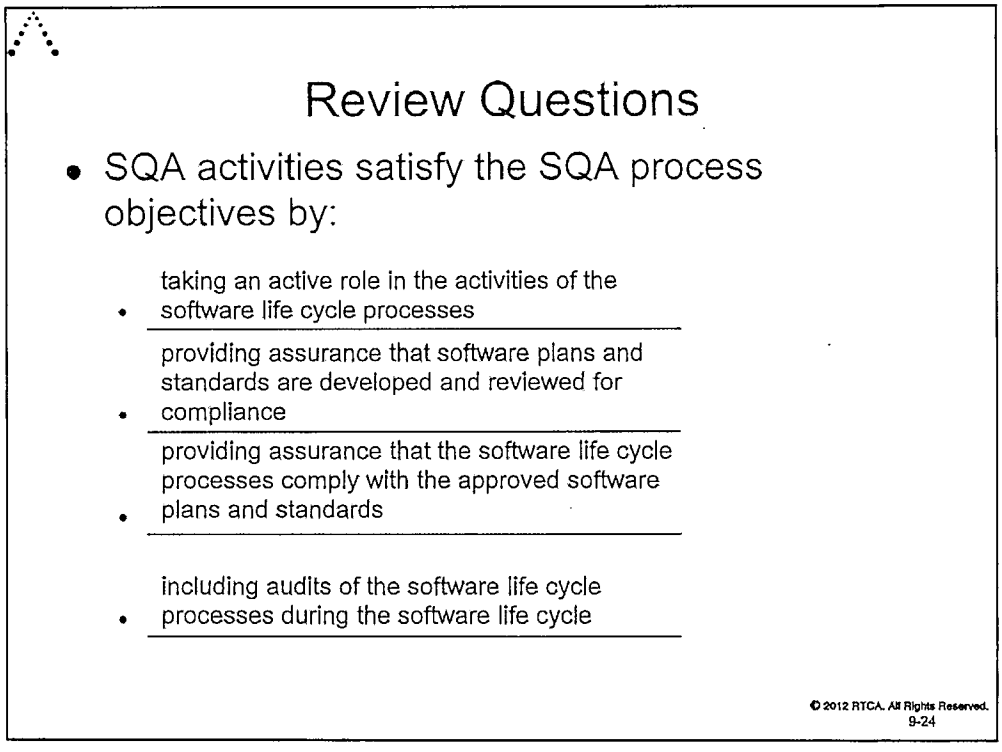
- The SQA process outputs in Table A-9 are ___?

A. CC1

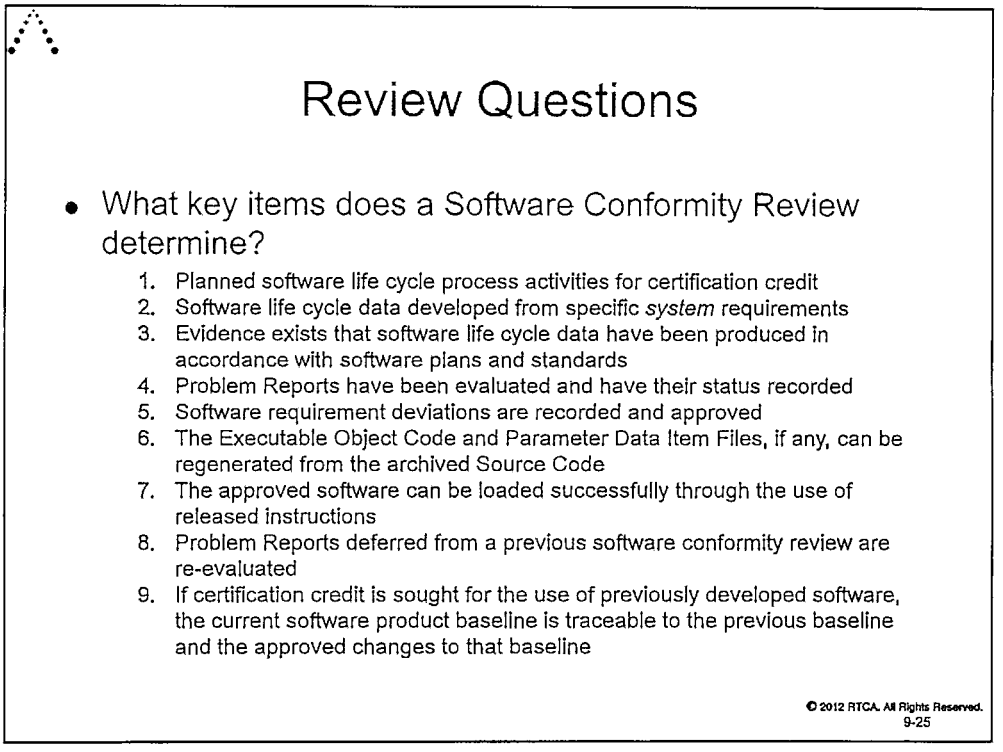
B. CC2 

9-23

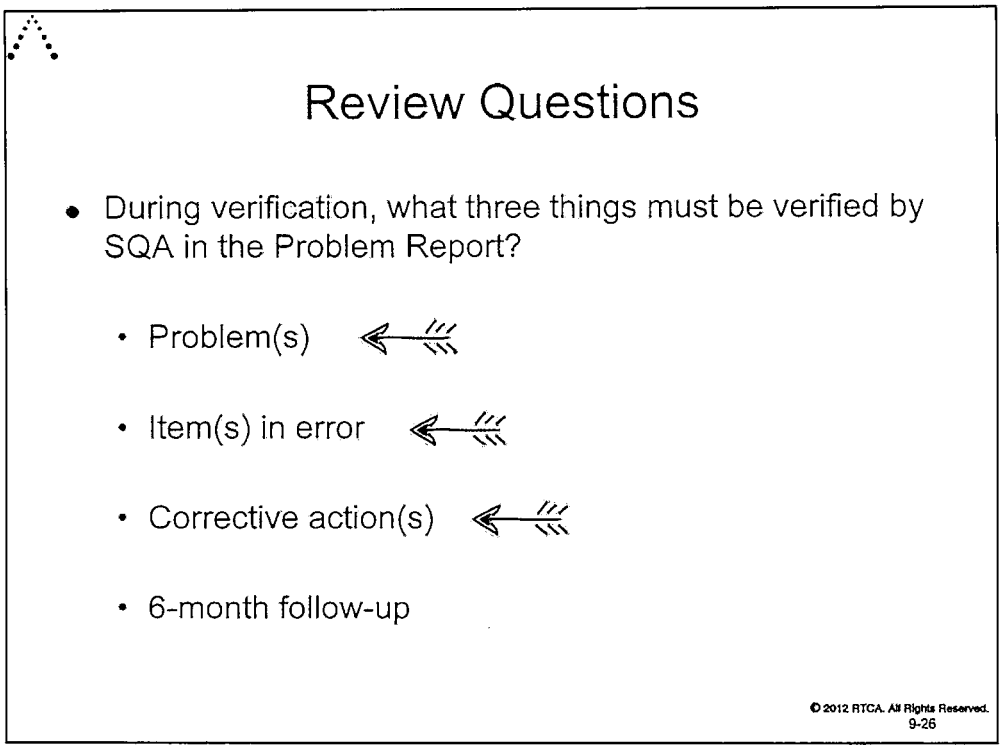
B. CC2



1. Take an active role in the activities of the software life cycle processes, and have those performing the SQA process enabled with the authority, responsibility, and independence to ensure that the SQA process objectives are satisfied.
2. Provide assurance that software plans and standards are developed and reviewed for compliance with this document and for consistency.
3. Provide assurance that the software life cycle processes comply with the approved software plans and standards.
4. Include audits of the software life cycle processes during the software life cycle to obtain assurance.



- Planned software life cycle process activities for certification credit, including the generation of software life cycle data, have been completed and records of their completion are retained.
- Software life cycle data developed from specific *system* requirements, safety-related requirements, or software requirements are traceable to those requirements.
- Evidence exists that software life cycle data have been produced in accordance with software plans and standards, and is controlled in accordance with the SCM Plan.
- Evidence exists that Problem Reports have been evaluated and have their status recorded.
- Software requirement deviations are recorded and approved.
- The Executable Object Code and Parameter Data Item Files, if any, can be regenerated from the archived Source Code.
- The approved software can be loaded successfully through the use of released instructions.
- Problem Reports deferred from a previous software conformity review are re-evaluated to determine their status.
- If certification credit is sought for the use of previously developed software, the current software product baseline is traceable to the previous baseline and the approved changes to that baseline.

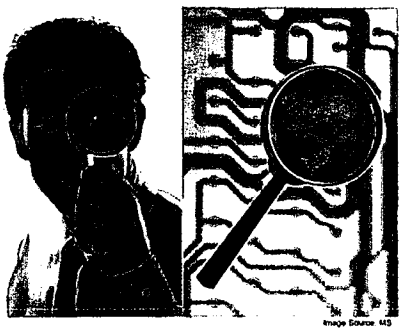


The answer is the first three items.

The 6 month follow-up is to make sure the Corrective Actions got accomplished,

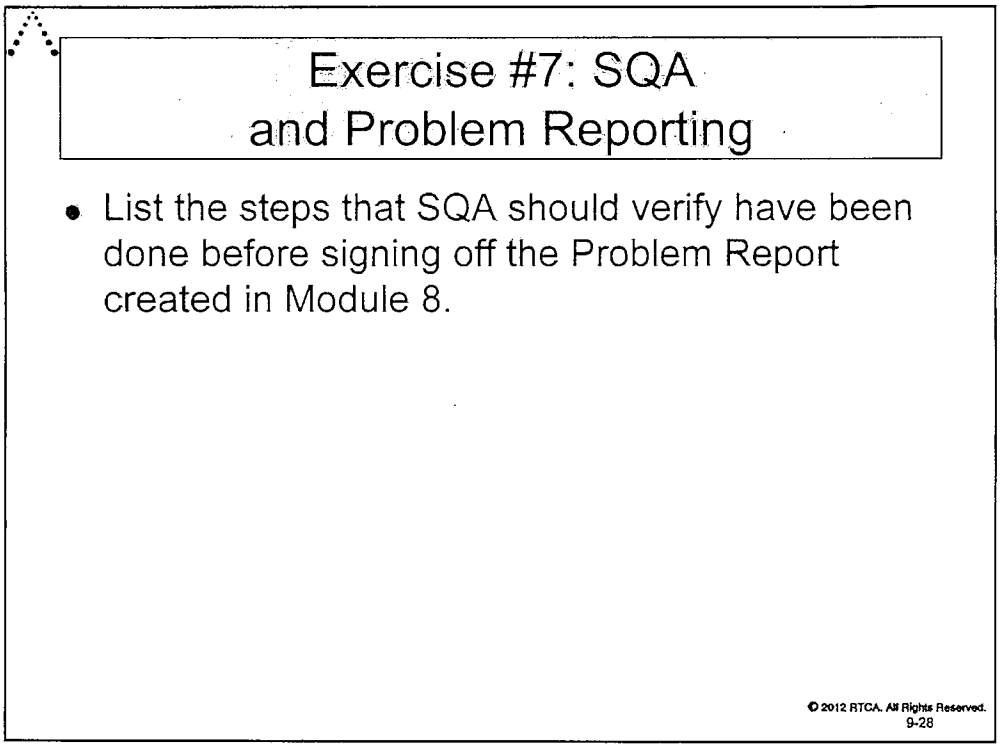
What steps should be done to close a Problem Report?

Class Discussion



Possible items to be done:

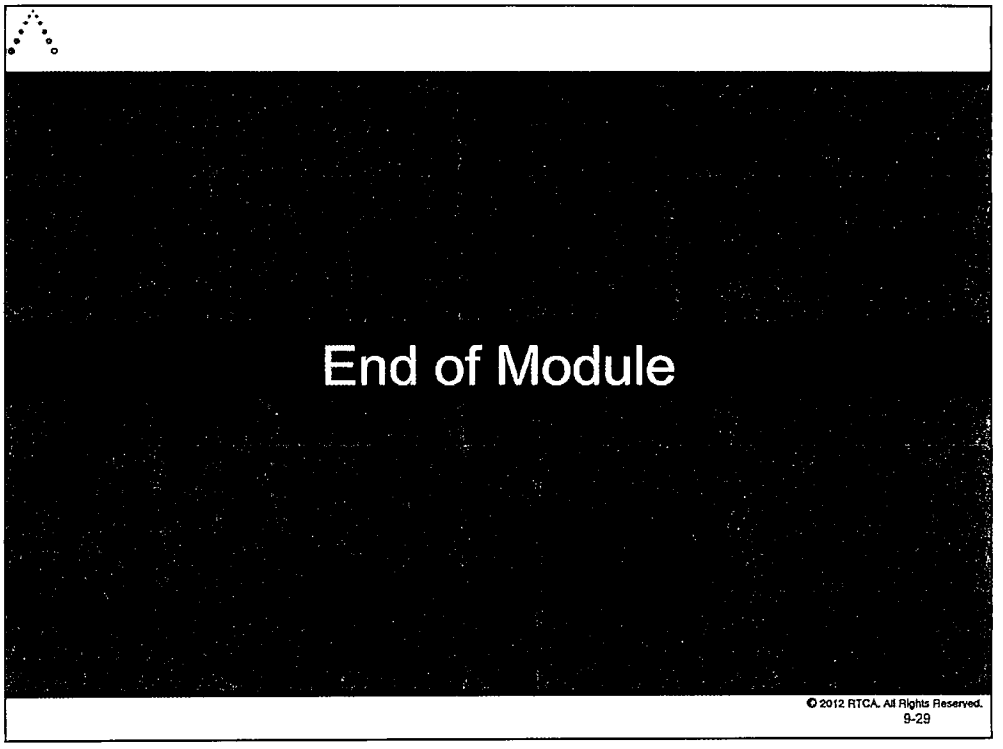
- The Source Code has been changed
  - The version must have been updated
  - The change be visible it the Source Code from the previous version to the current version
- Document the change in the problem report
- Re-review the Source Code
- Document the identification of the re-review results in the problem report
- Get agreement that the problem is resolved from the appropriate Software personnel (For Example the team lead)



Have the class brainstorm a list and put it on a white board on chart.

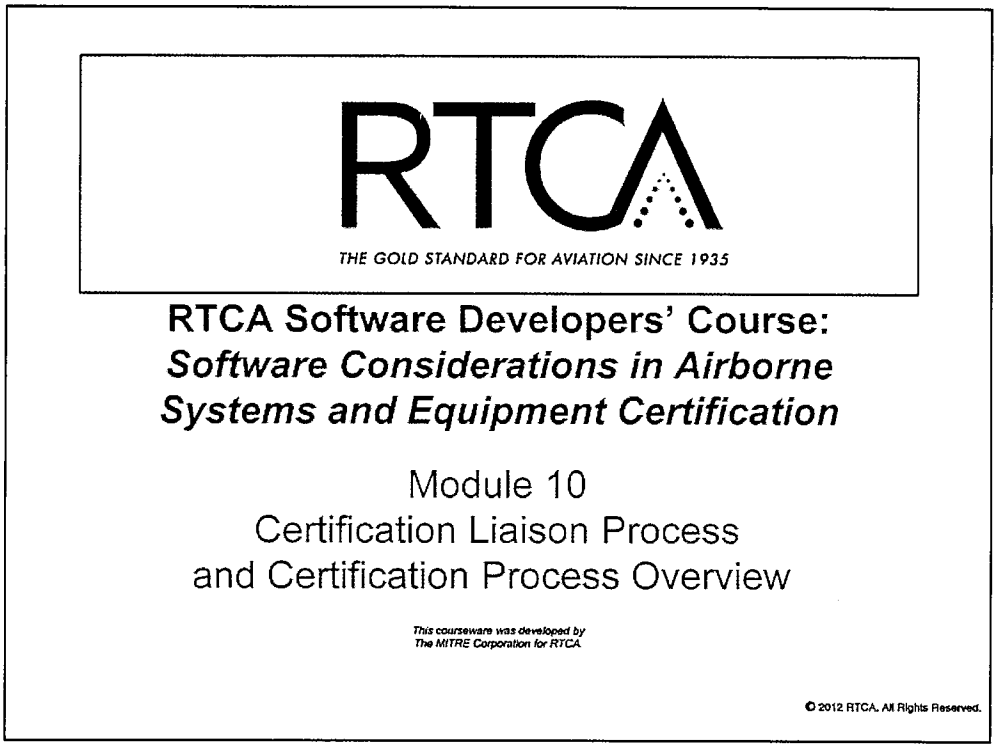
Possible items to be done:

- The Source Code has been changed
    - The version must have been updated
    - The change be visible it the Source Code from the previous version to the current version
- Document the change in the problem report
- Re-review the Source Code
- Document the identification of the re-review results in the problem report
- Get agreement that the problem is resolved from the appropriate Software personnel (For Example the team lead)

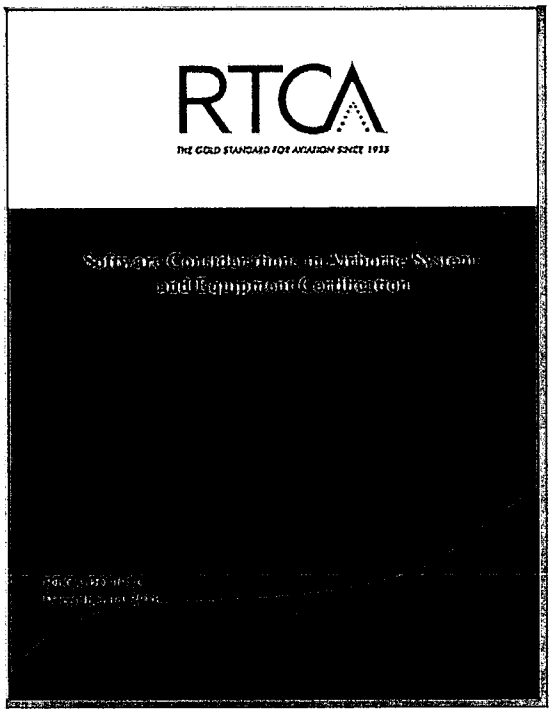

End of Module
© 2012 RTCA. All Rights Reserved.
9-29

Exercise # 7
SQA and Problem Reporting

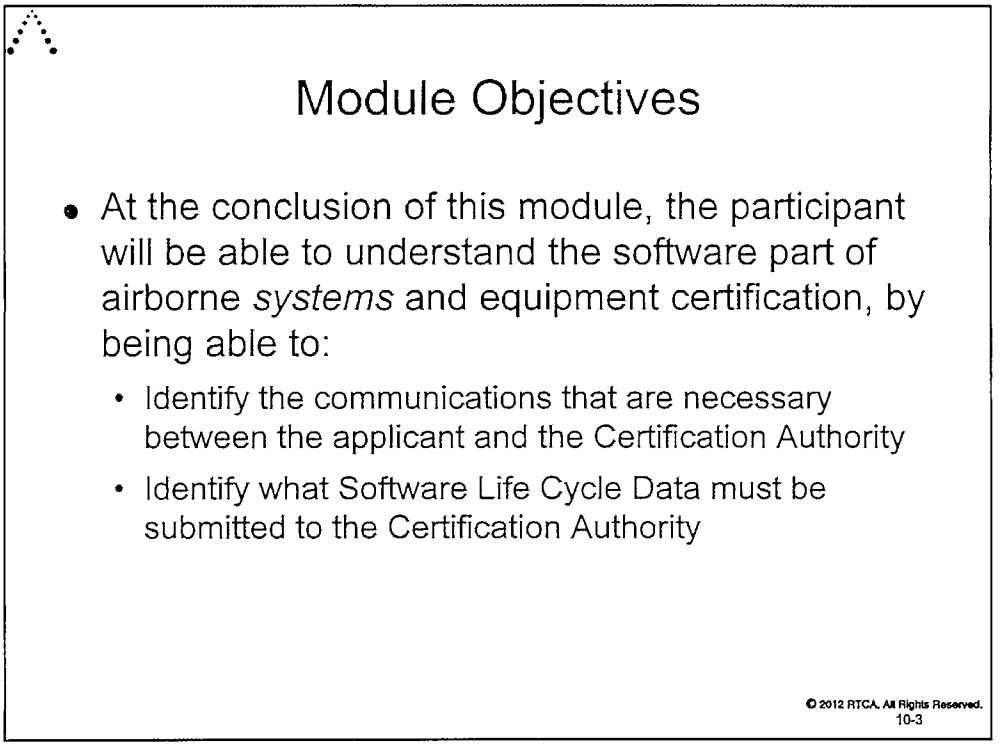
List the steps that SQA should verify have been done before signing off the Problem Report created in Module 8.



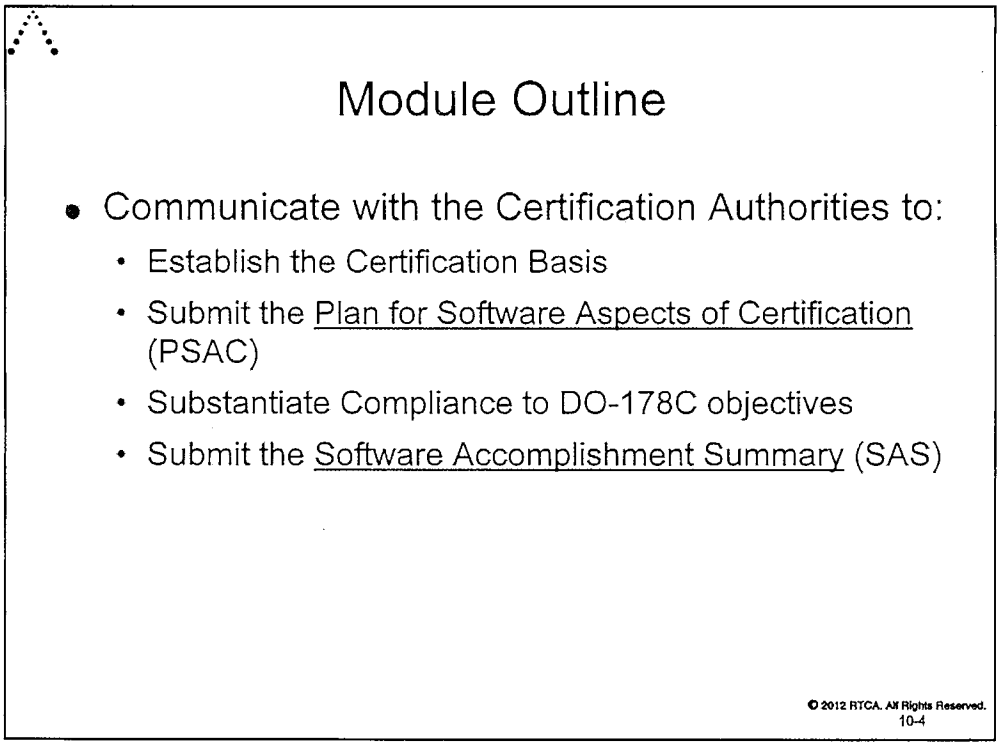
This 1-hour module introduces the Certification Liaison process to establish communication and understanding between applicants and Certification Authority throughout the software life cycle to assist the certification process, gain an agreement on means of compliance through approval of Plan for Software Aspects of Certification (PSAC), and provide compliance substantiation. For this process, objectives, outputs and activities are described. It also includes certification process overview for, reviewing the basis, Certification Authority assessment of plan for software aspects of certification for completeness and consistency and determination that the product complies with the certification basis.



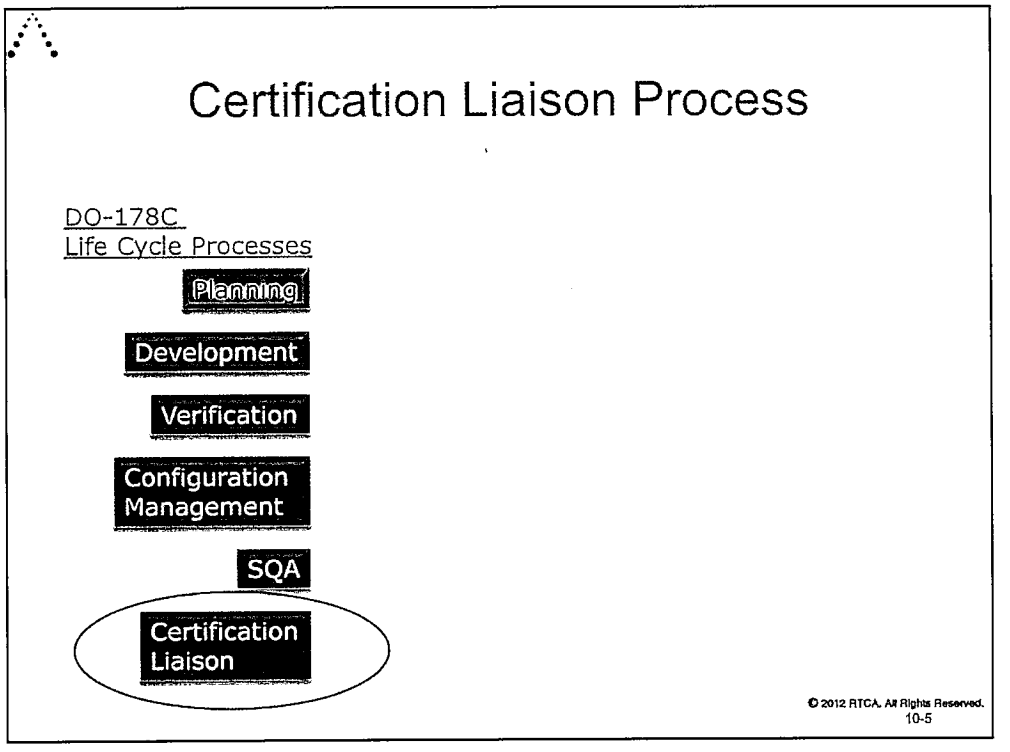
Sections 9 & 10



- Certification is a legal action done by the Certification Authority for airborne *systems* or equipment that is compliant to the required certification basis.
  - Software is considered a *component* within the airborne *system* or equipment.
  - Software, in and of itself, is not certified or approved
- The applicant is the person or organization seeking a certificate and/or approval from the Certification Authority
- DO-178C is applied to software for:
  - TC – Type Certificate
  - STC – Supplemental Type Certificate
  - ATC – Amended Type Certificate
  - TSOA – Technical Standard Order Approval (not a certificate but and approval)



- The Certification Basis are the rules and regulations that the applicant must comply with to achieve certification. DO-178C is not a rule or regulation but a means to achieve compliance with, for example, Title 14 CFR part 25.1309. The PSAC will document the agreed certification basis for the *system's* software.
- The PSAC will be covered in a following slide.
- Substantiate compliance, from the Certification Authority point of view, is done by audits and/or reviews of evidence (software life cycle data in Chapter 11). From the applicant point of view, it is supporting the audits and/or reviews by producing the evidence of compliance from the software life cycle processes and making it available to the certification authority.
- The Software Accomplishment Summary is where the applicant states that compliance to DO-178C has been achieved. The certification Authority either approves the evidence of compliance, agreeing with the applicant's statement of compliance, or does nothing. *Systems* are not dis-approved, they are just not approved.



Three sub-processes in the software planning process are:

- Plan SW Development and Integral Processes
- Develop Software Standards
- Review Plans and Standards

Discuss "activities": Use backup slide defining "activities"

Plans, when followed, need to produce products compliant to DO-178C objectives.

Certification Basis

- The Certification Basis is established by the *system* processes during certification planning
- Software Level, which has a dependency on the Certification Basis, is what the *system* processes give the software processes

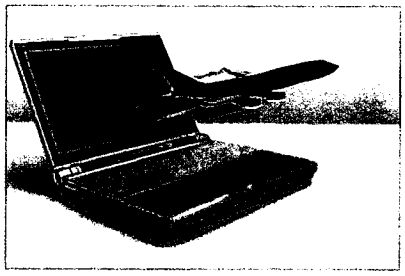


Image Source: MS

10-6

- This is covered here because it is mentioned in DO-178C sections 9 and 10 and is very important at the *system* level.
- The Certification Basis is established at a *system* level; and what the software sees is at the *software assurance level*. The rules and regulations that the applicant must comply with to achieve certification. DO-178C is not a rule or regulation but a means to achieve compliance with, for example, Title 14 CFR part 25.1301. The PSAC will document the agreed certification basis for the *system's* software as the software assurance level.
- The only part of the Certification Basis that the Software Life Cycle Processes see is the Software Assurance Level.
- Sample Certification Basis for FAA Document 8110.4c:

b. The Certification Basis is defined as:

(1) Title 14 CFR part 25, dated February 1, 1965, with Amendments 1 through 22

"Airworthiness Standards: Transport Category Airplanes," and 14 CFR § 25.471 of Amendment 25-23,

for all areas not affected by the change.

(2) Title 14 CFR part 25, dated February 1, 1965, including Amendments 25-1 through 25-89

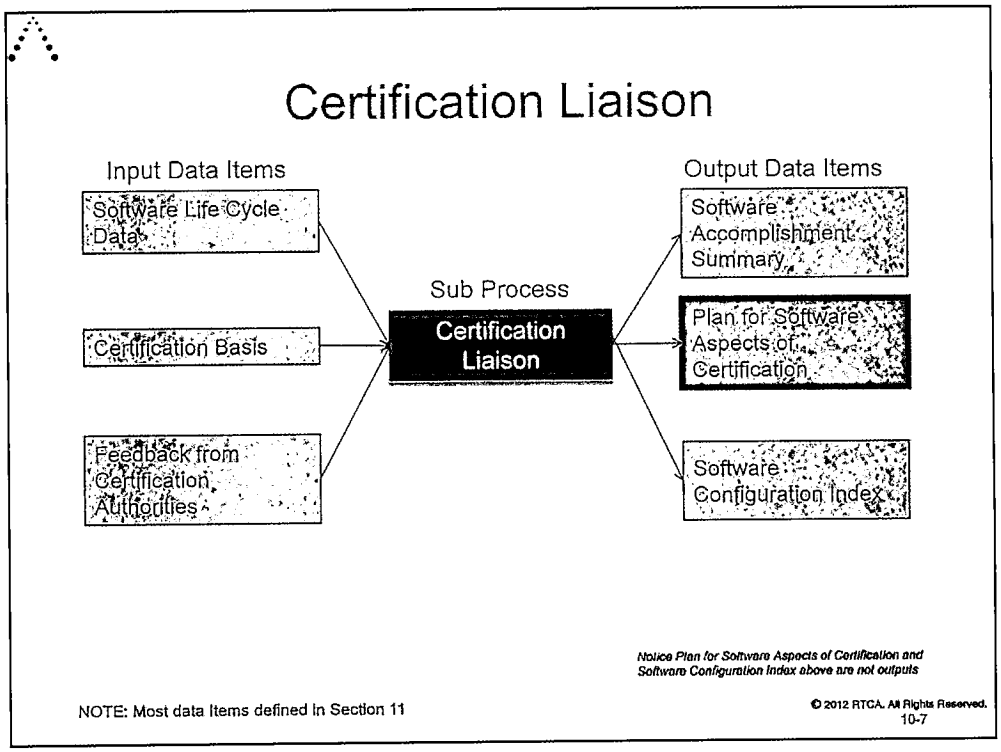
for the change and all areas affected by the change. The following lists the Federal Aviation Regulations

complied with through Amendment Level 25-89.

25. 125 25. 605 25. 685 25. 841 25. 1039 25. 1326 25. 1435 25. 1541

25. 145 25. 607 25. 689 25. 843 25. 1141 25. 1327 25. 1439 25. 1543

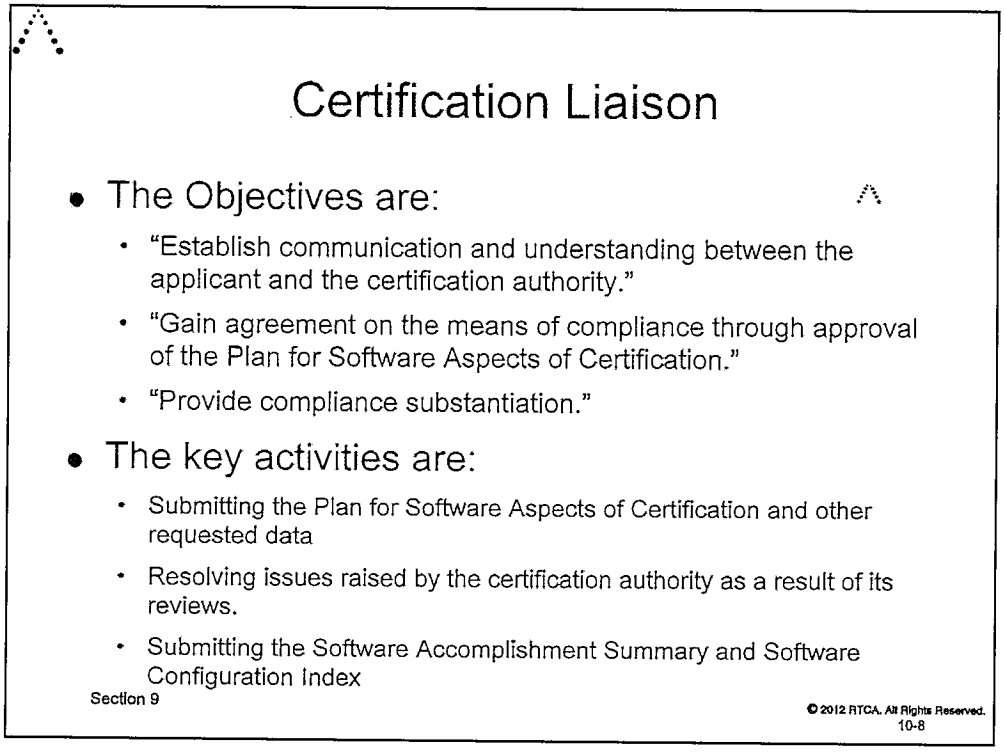
25. 149 25. 609 25. 693 25. 855 25. 1142 25. 1329 25. 1441 25. 1545



The Certification Liaison Process provides the communications between the applicant and the certification authorities.

Inputs are all Software Life Cycle Data except for the SAS. Additionally, the Certification Basis and feedback from the Certification Authority are inputs.

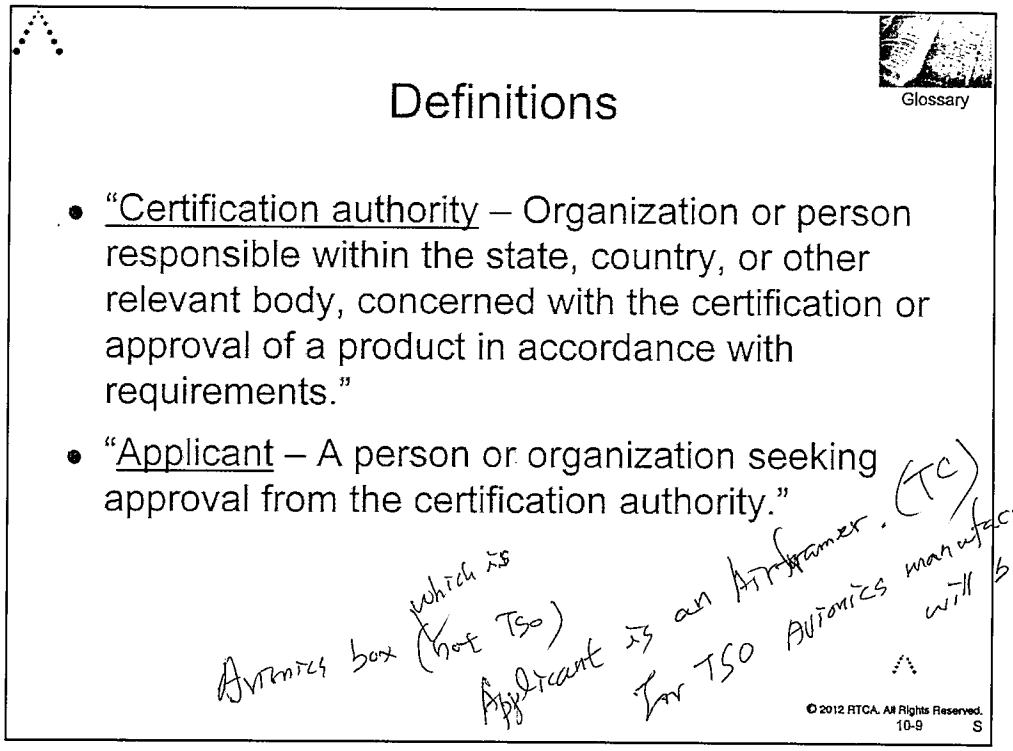
This process produces the Software Accomplishment Summary. It also supplies at least the PSAC and SCI (produced in other processes) to the Certification Authority.



Communication on software starts with the PSAC. It should be submitted before most of the Software Life Cycle processes start so that any feedback from the Certification Authority will not cause a lot of re-work.

Compliance Substantiation means submittal of requested life cycle data as well as support of any audits/reviews by the Certification Authority.

Communication with the Certification Authority through the software life cycle is important because any issues that are raised can then be dealt with early before there is a need for a lot of re-work. For example: get an audit/review on planning before you start the development process if possible.



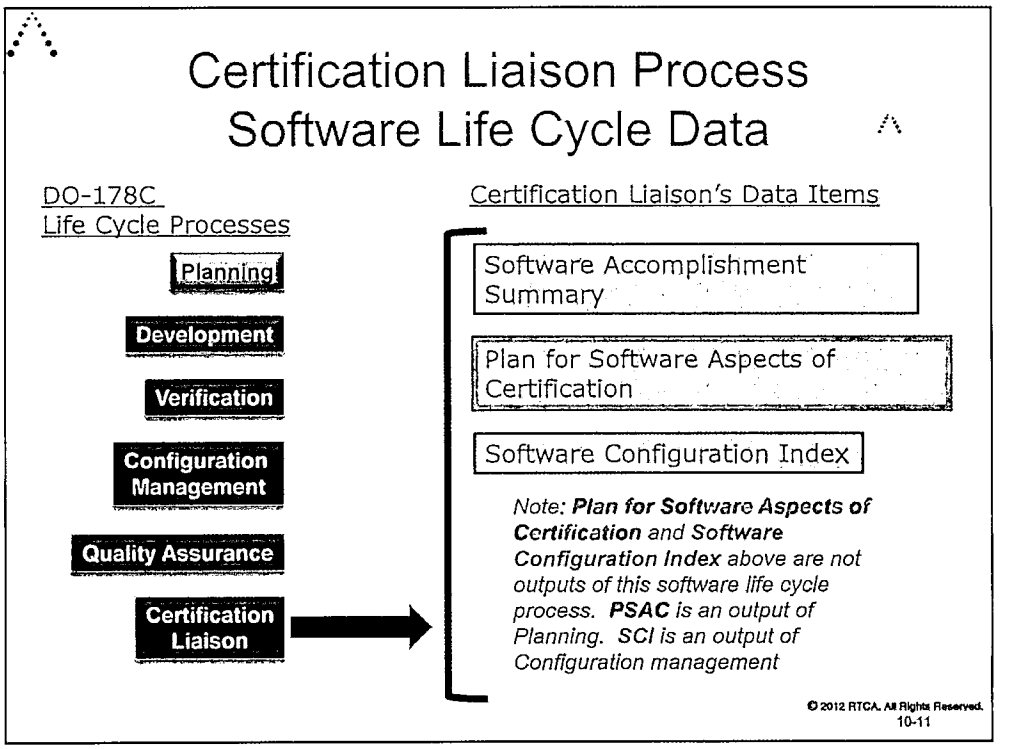
The FAA AVS is an example of a Certification Authority in airborne systems. They have legal authority in the US. Europe has another certification Authority.

The manufacturer of the avionics is an example of an Applicant.

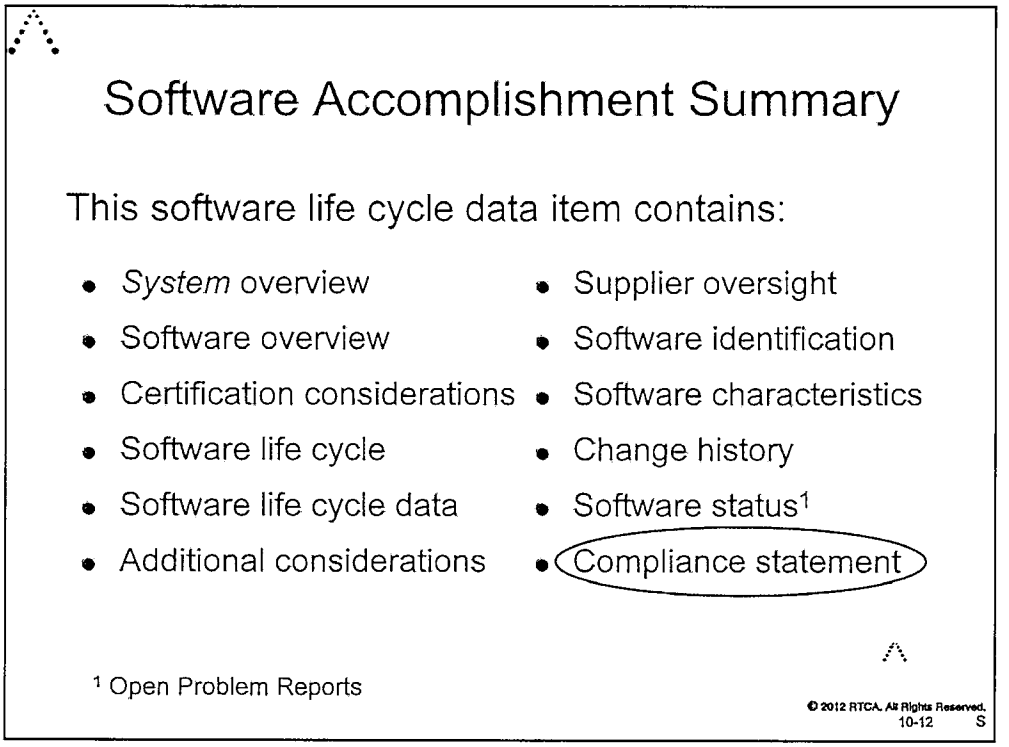
Glossary

Definitions (cont'd)

- "Approval – The act or instance of giving formal recognition or official sanction."
- "Certification – Legal recognition by the certification authority that a product, service, organization, or person complies with the requirements."
- "Means of compliance – The intended method(s) to be used by the applicant to satisfy the requirements stated in the certification basis for an aircraft, engine, propeller, or, by region, auxiliary power unit."



1. Software Accomplishment Summary is produced by this process.
2. Plan for Software Aspects of Certification is produced during the Planning Process and used by the Certification Liaison Process.
3. Software Configuration Index is produced during the Configuration Management Process
4. Plan for Software Aspects of Certification, Software Accomplishment Summary, and Software Configuration Index are the minimum software life cycle data submitted to the Certification Authority
5. "Type design – For the purposes of this document, the type design consists of the following: (1) The drawings and specifications, and a listing of those drawings and specifications, necessary to define the configuration and the design features of the certified product shown to comply with the requirements for that product; (2) Any other data necessary to allow, by comparison, the determination of the airworthiness of later products of the same type."
6. Type Design Data is the software life cycle data requiring approval by the Certification Authority
    1. Any of this data may have to be submitted to the Certification Authority
    2. This data must be available to the Certification Authority for 5 years beyond last use in an aircraft



- While not *required*, it is a good idea that this is a stand alone document.
- Has a lot of the same sections as the PSAC but is done at the end of the product development and not at the beginning.
- Must have a list of all open problems at time of certification
- Documents what actually happened.
- It is considered a statement of compliance by the applicant.
- Need to discuss the problem of referencing the SAS from within its self and the need to complete it last and how that affects the conformity review of SQA.
- Is used by the Certification Authority to accept the software as compliant to the objectives of DO-178C

Plan for Software Aspects of Certification

- This plan should be written near the beginning of a product development effort
- Certification Authority agreement with this plan should be obtained as soon as possible after its completion

10-13 S

The Plan for Software Aspects of Certification (PSAC) starts the certification process for software with the Certification Authority.

The Plan for Software Aspects of Certification (see 11.1) serves as the primary means for communicating the proposed software life cycle processes to the certification authority for agreement.

Plan for Software Aspects of Certification (cont'd)

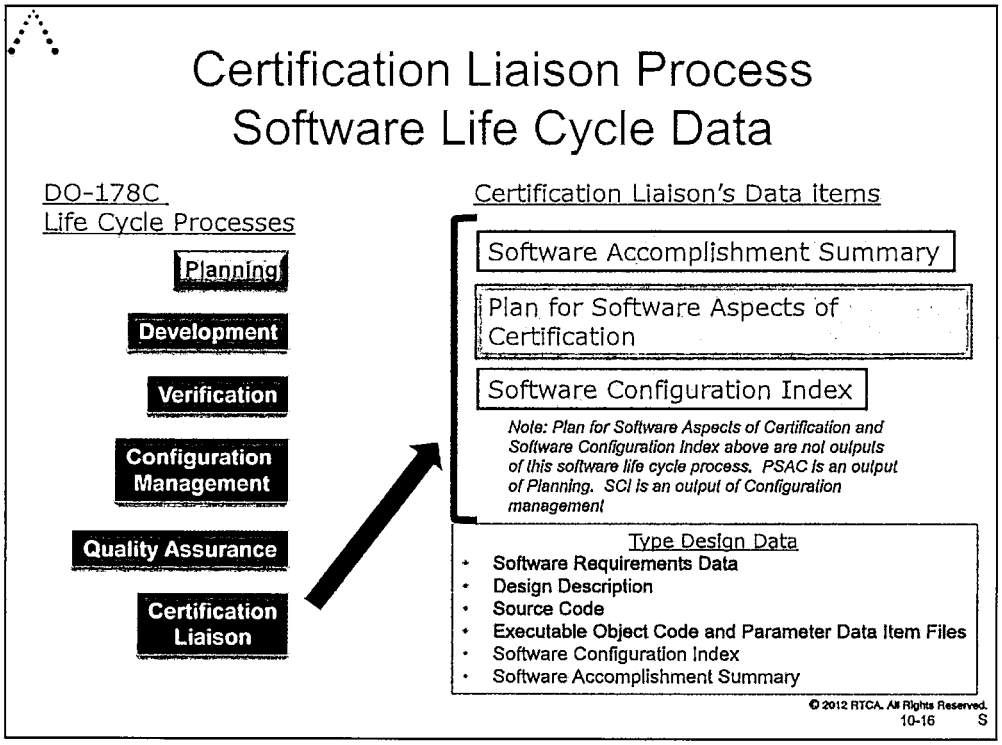
- Plan contains the following eight items:
  - *System* overview
  - Software overview
  - Certification considerations
    - Software Assurance Level as determined by the *system* safety process
  - Software life cycle
  - Software life cycle data
  - Schedule
  - Additional considerations
  - Supplier oversight

10-14 S

- While not ***required***, it is a good idea that this is a stand alone document.
- See Module 5 and 11 for more information on the PSAC.

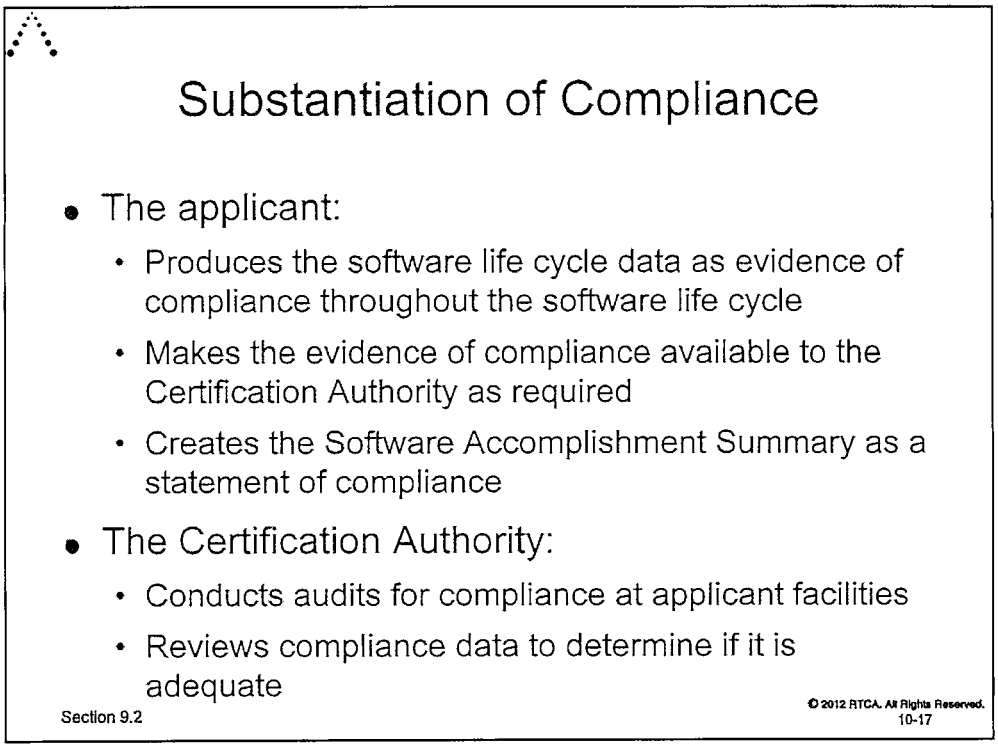
Software Configuration Index (SCI)

- SCI provides specific configuration and version identifiers for:
  - Software product including previously developed software
  - Executable Object Code (with Data Integrity Checks) and Parameter Data Items Files with instructions to build them
  - Each source Code Component
  - Software Life Cycle Data
  - Archive and Release Media
  - Procedures, Methods and Tools for making modifications to user-modifiable software
  - Procedures and methods for loading software into target hardware



1. Software Accomplishment Summary is produced by this process.
2. Plan for Software Aspects of Certification is produced during the Planning Process and used by the Certification Liaison Process.
3. Software Configuration Index is produced during the Configuration Management Process
4. Plan for Software Aspects of Certification, Software Accomplishment Summary, and Software Configuration Index are the minimum software life cycle data submitted to the Certification Authority
5. "Type design – For the purposes of this document, the type design consists of the following: (1) The drawings and specifications, and a listing of those drawings and specifications, necessary to define the configuration and the design features of the certified product shown to comply with the requirements for that product; (2) Any other data necessary to allow, by comparison, the determination of the airworthiness of later products of the same type."
6. Type Design Data is the software life cycle data requiring approval by the Certification Authority
    1. Any of this data may have to be submitted to the Certification Authority
    2. This data must be available to the Certification Authority for 5 years beyond last use in an aircraft

    This is a regulatory requirement for Part 121, determined by an interpretation. This list could change. It is not part of the PSAC.



- It is a good idea that the applicant will produce the evidence of compliance between producing the PSAC and the SAS. The FAA may say that upon achieving a program milestone, they may do an audit.
- Any audits by the Certification Authorities should be staged throughout the Software Life Cycle. For examples audits involving requirements maybe done after requirements are completed (they may later need to be changed some) and before coding is very far along. The FAA recognizes 4 Stages of Involvement (SOI) in it Job Aid (get from the FAA by asking your Aircraft Certification Office (ACO) engineer). These stages are named as:
    - Planning
    - Requirements, Design, and Coding
    - Verification
    - Final (product is finished) – the final audit **may** get done just ahead of completing the SAS with a review of it after it is completed.
- Reviews of compliance data by the FAA could generally follow the same stages as the audits. However only "reviews" are done on simple programs with low assurance levels while reviews and audits are both done on complex programs and high assurance levels.

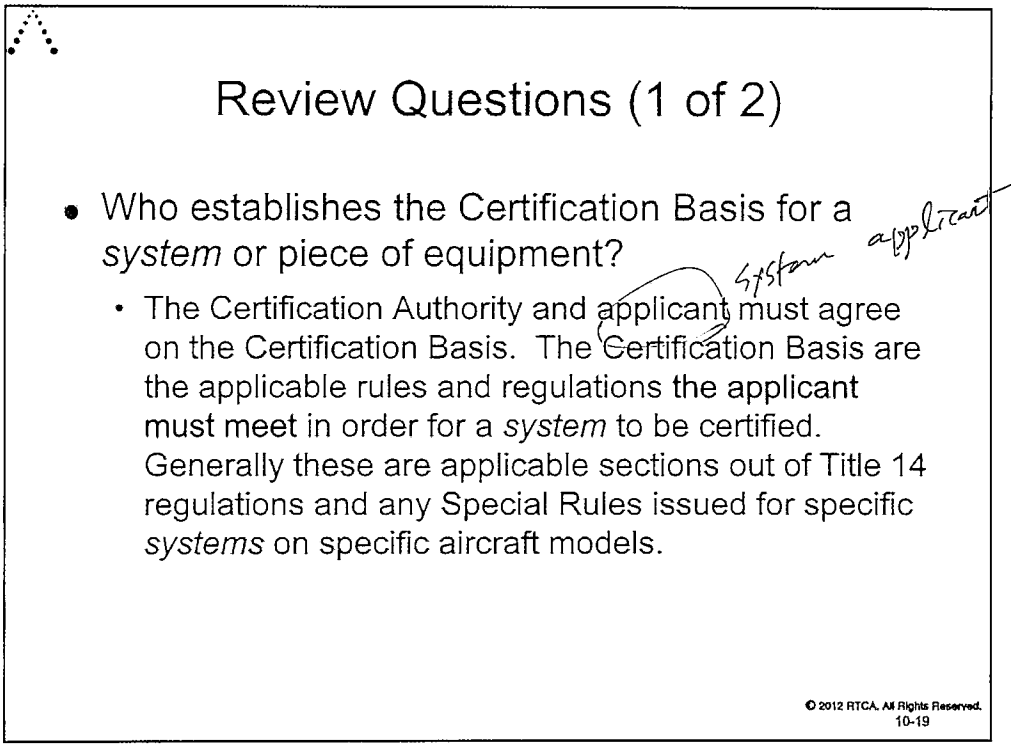
Software Life Cycle Data Submitted to the Certification Authority

- Must be submitted:
  - Plan for Software Aspects of Certification
  - Software Configuration Index
  - Software Accomplishment Summary
- Certification Authority may want one or more of the following submitted:
  - Software Requirements Data
  - Design Description
  - Source Code
  - Executable Object Code and Parameter Data Item Files

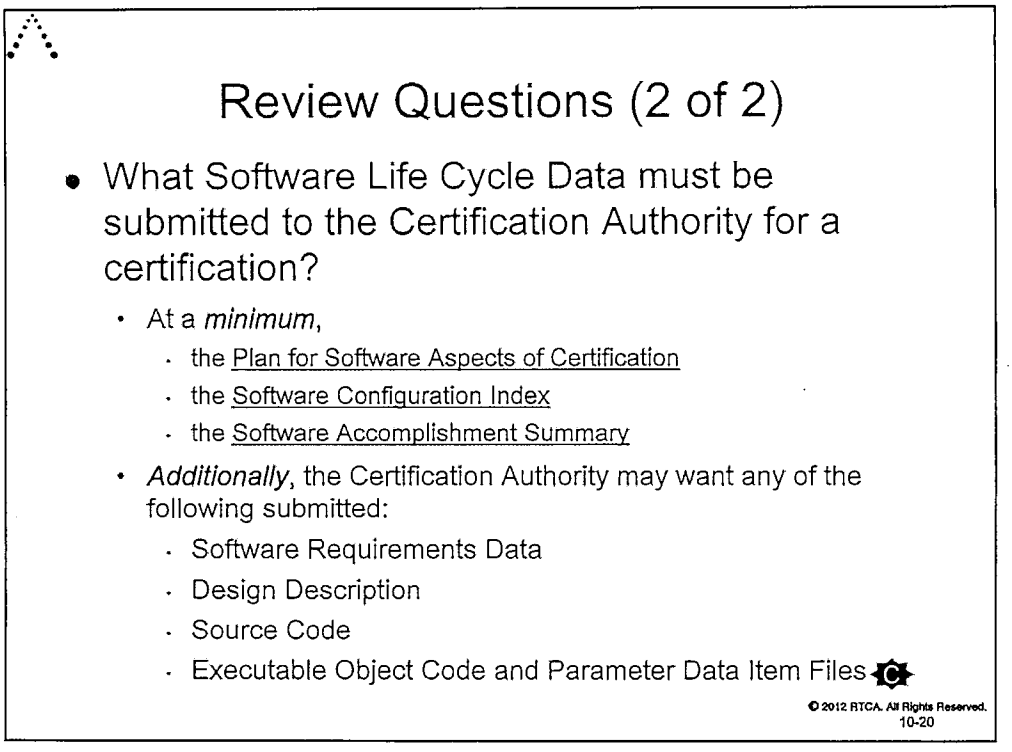
Software Life Cycle Data Archiving should be planned so that the data is available and readable for many years.

Section 9.3

- All this Software Life Cycle Data is considered Type Design Data
- Certification Authorities will want to keep a copy of the "must be submitted" items.
  - These items may be submitted electronically at the discretion of the Aircraft Certification Office and in a format agreed to
- The "may want" items at the very least need to be available for review.
- All Software Life Cycle data need to be available in a readable manner until 5 years after last use of the certified airborne *system* or equipment. Verification results might be very important if there are problems later with the *system* or equipment.

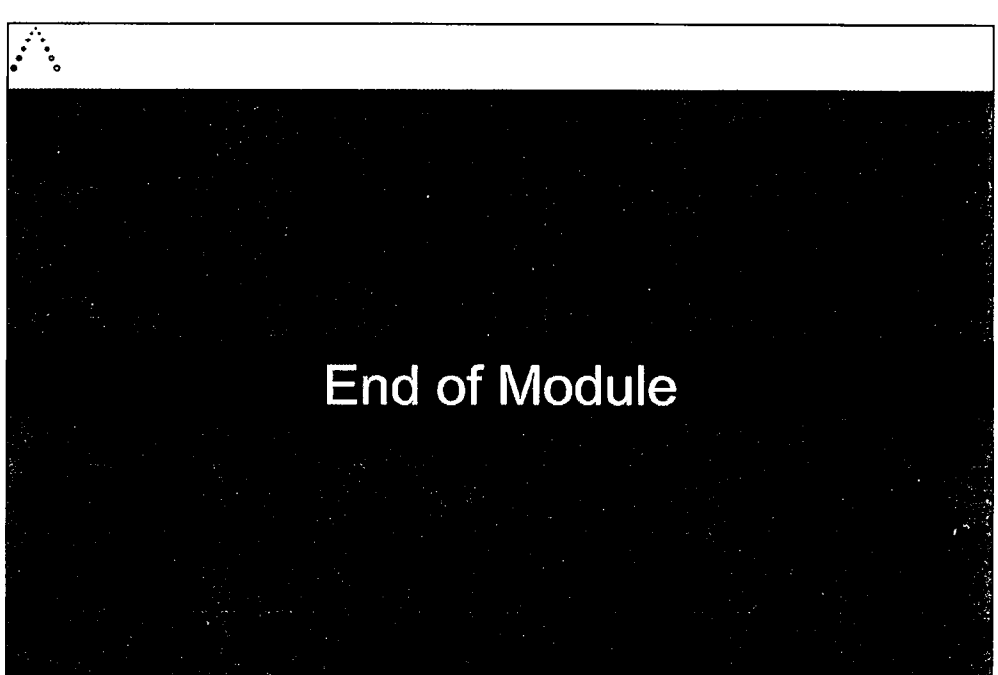


Question 1: The Certification Authority and applicant must agree on the Certification Basis. The Certification Basis are the applicable rules and regulations the applicant must meet in order for a *system* to be certified. Generally these are applicable sections out of Title 14 regulations and any Special Rules issues or issued? for specific *systems* on specific aircraft models.

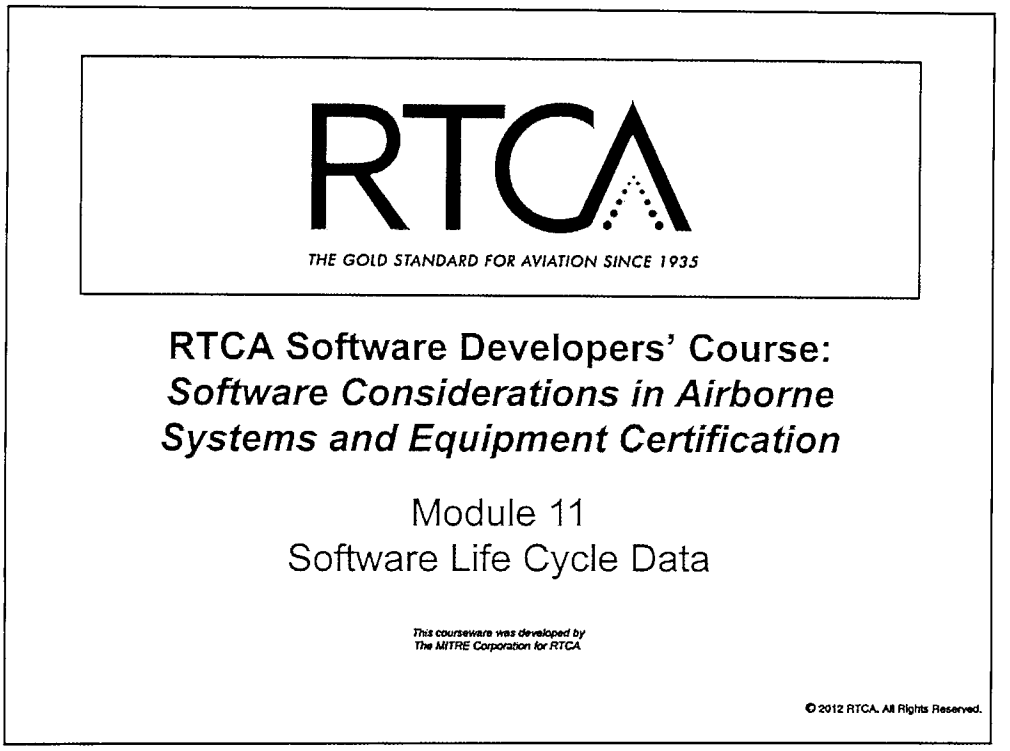


Question 2: At a minimum, the Plan for Software Aspects of Certification, the Software Configuration Index, and the Software Accomplishment Summary must be submitted. Additionally, the Certification Authority may want any of the following submitted:

- Software Requirements Data.
- Design Description.
- Source Code.
- Executable Object Code and Parameter Data Item Files. (new to DO-178C)

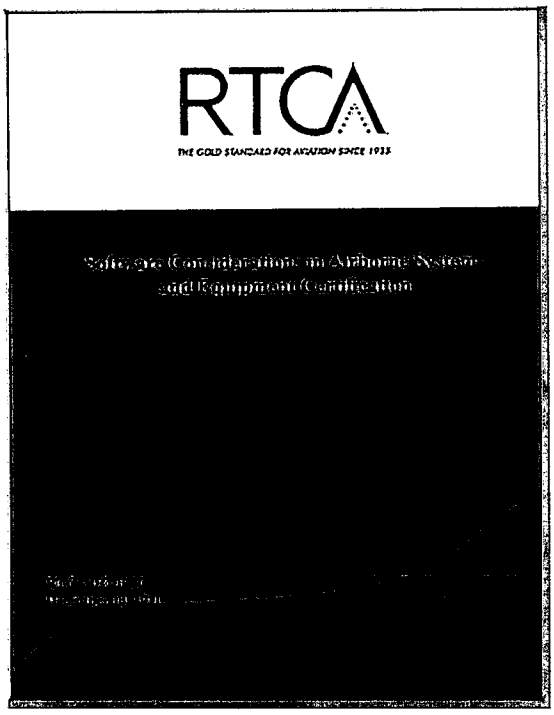


10-21

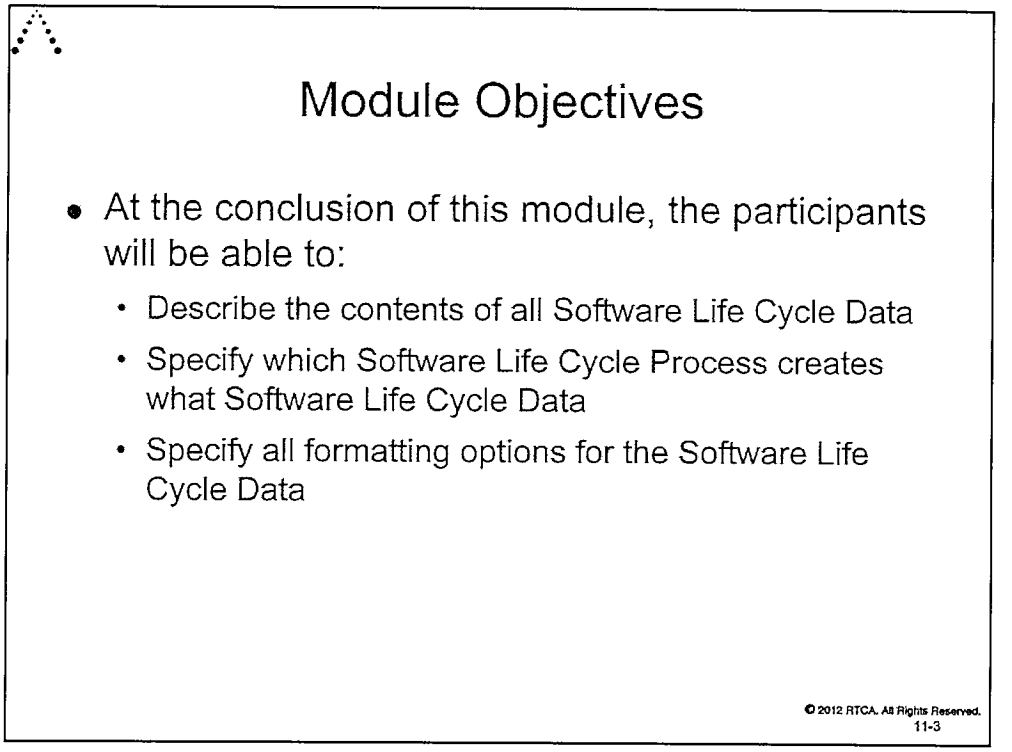


This 1-hour module presents a detailed look at software life cycle data: its characteristics, form, configuration management controls and content. The module describes the contents of all Software Life Cycle Data, and provides a clear understanding of which Software Life Cycle Process creates what specific Software Life Cycle Data.

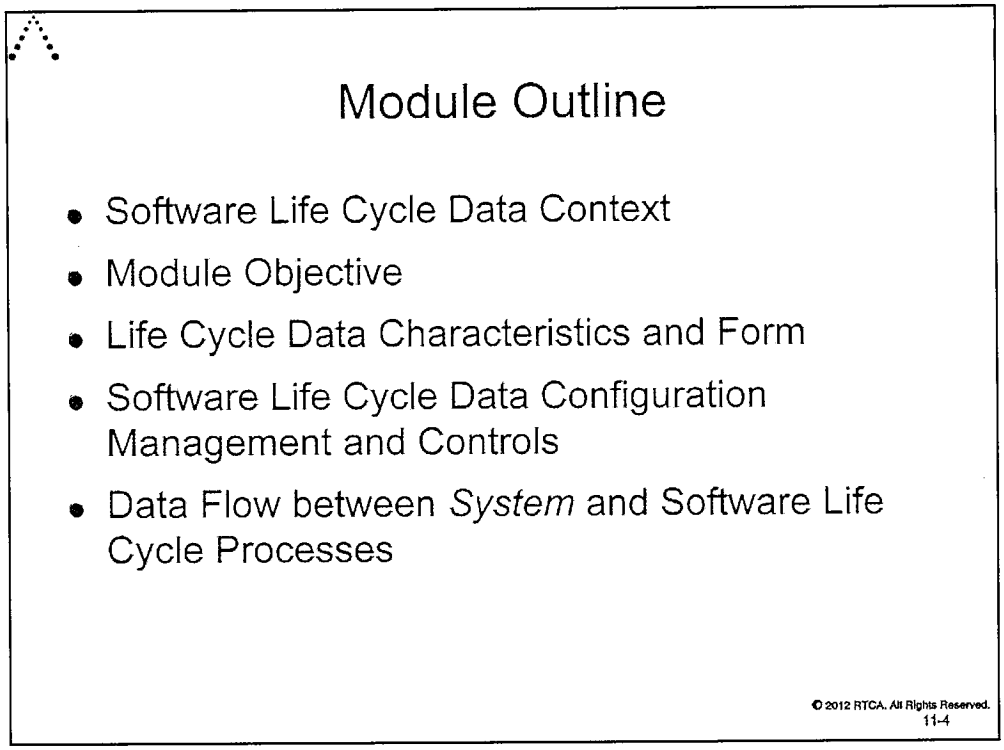
A detailed diagram of the flow of data from the processes is used to re-enforce earlier presentations.



Section 11



This chart identifies the module 11 objectives.



This chart and the next three charts provides an outline of Module 11.

Module Outline (Continued)

- Planning Processes and Data Items
  - Plan for Software Aspects of Certification
  - Software Development Plan
  - Software Verification Plan
  - Software Configuration Management Plan
  - Software Quality Assurance Plan
  - Software Requirements Standard
  - Software Design Standard
  - Software Coding Standard
  - Software Verification Results
  - Software Requirements Data
  - Design Description

Module Outline (Concluded)

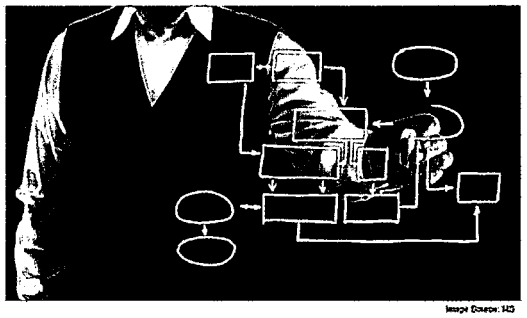
- Planning Processes and Data Items (Concluded)
  - Source Code
  - Executable Object Code
  - Trace Data
  - Parameter Data Item
  - Software Verification Cases and Procedures
  - Software Configuration Index
  - Software Life Cycle Environment Configuration Index
  - Problem Reports
  - Software Configuration Management Records
  - Software Quality Assurance Records
  - Software Accomplish Summary

Software Life Cycle Data Context

- Data is created to support software aspects of the certification liaison process
- Software life cycle processes produce data to plan, direct, explain, define, record evidence of activities
- Data enables software life cycle processes, *system*/equipment certification and post-certification modification of software product

Software Life Cycle Data Characteristics

- Unambiguous
- Complete
- Verifiable
- Consistent
- Modifiable
- Traceable



Section 11a

11-8

Unambiguous: Permits single interpretation

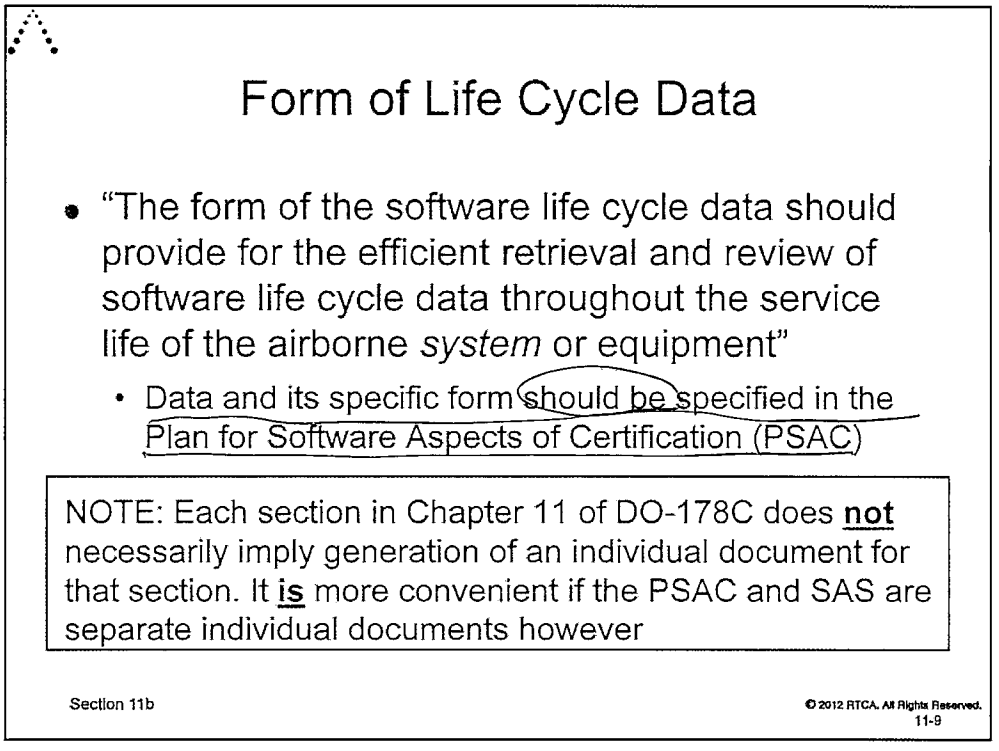
Complete: Includes necessary and relevant requirements and descriptive material

Verifiable: Can be checked for correctness by person/tool

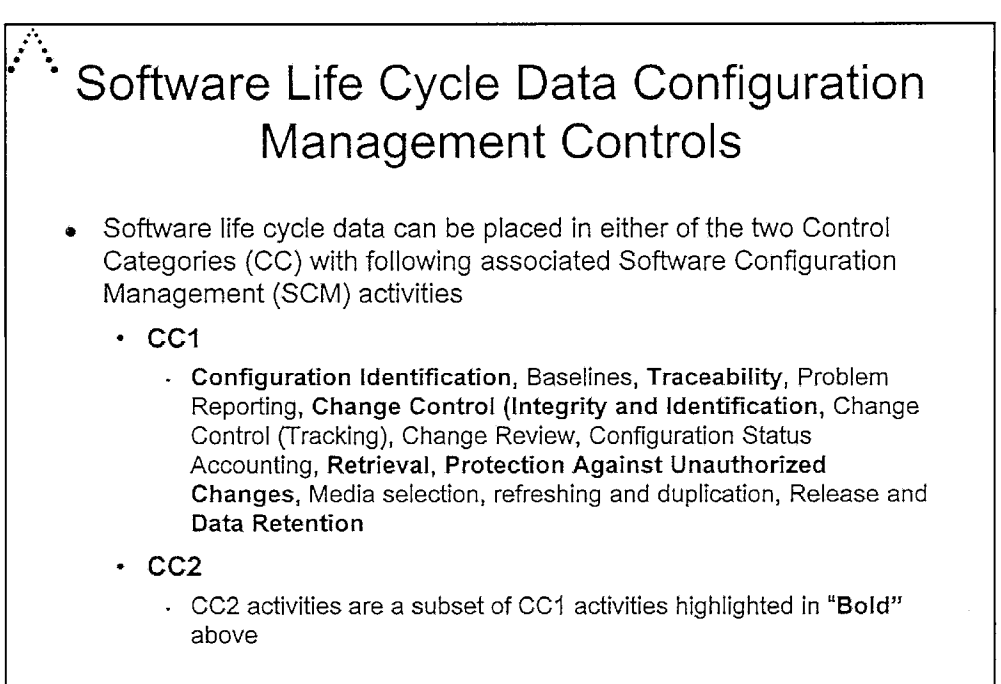
Consistent: Contains no conflicting information

Modifiable: Structured to permit complete, consistent and correct changes

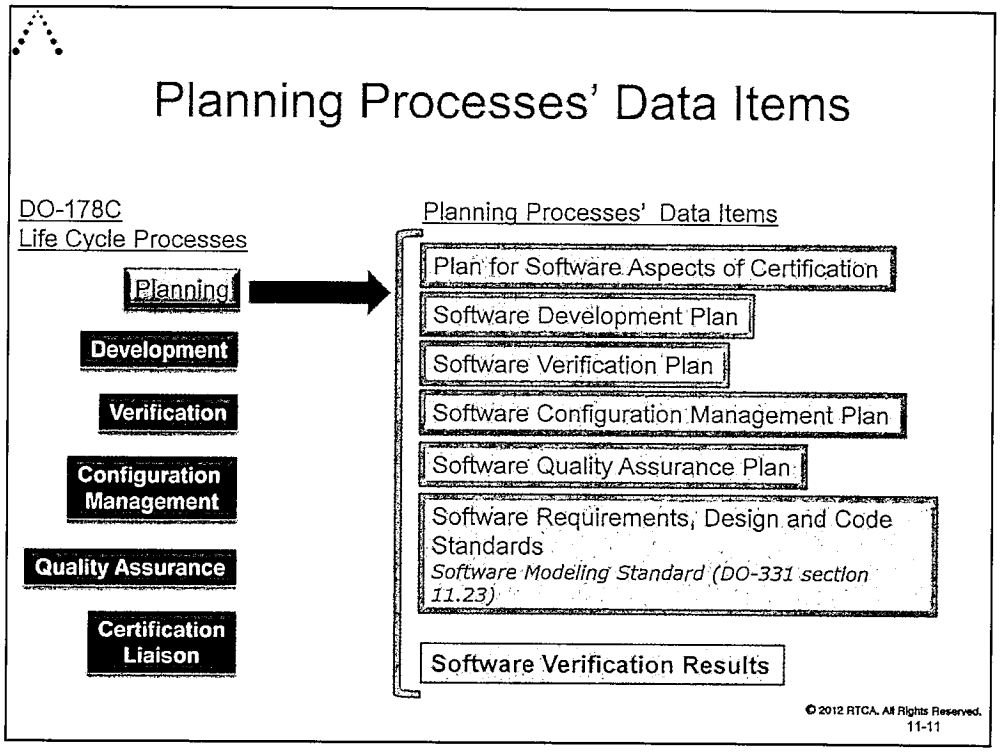
Traceable: Permits determination of components' origin



The PSAC should relate the Software Data Life Cycle Data Items to specific documents the applicant plans to create. This should also be included in the SAS at the end of the program.

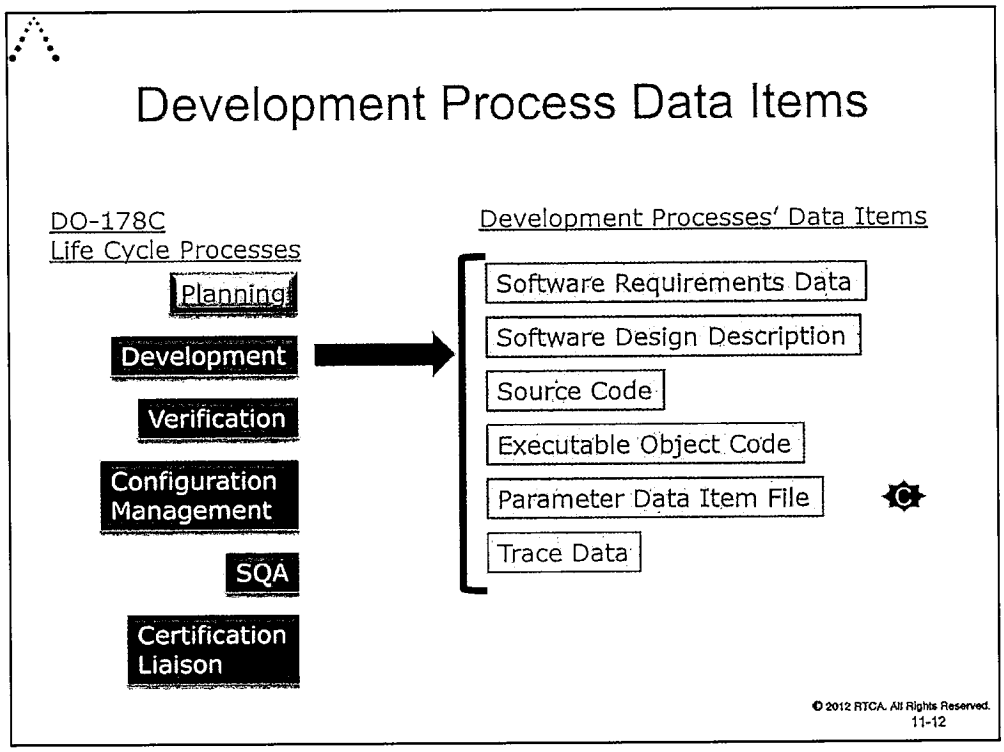


The control categories for Software Life Cycle data are specified in the Annex A tables. The CC1 is acceptable for data in CC2, however the applicant should do so with care, for example problem reports must be CC2, since a CC1 problem report would have to have a new problem report opened to close it.

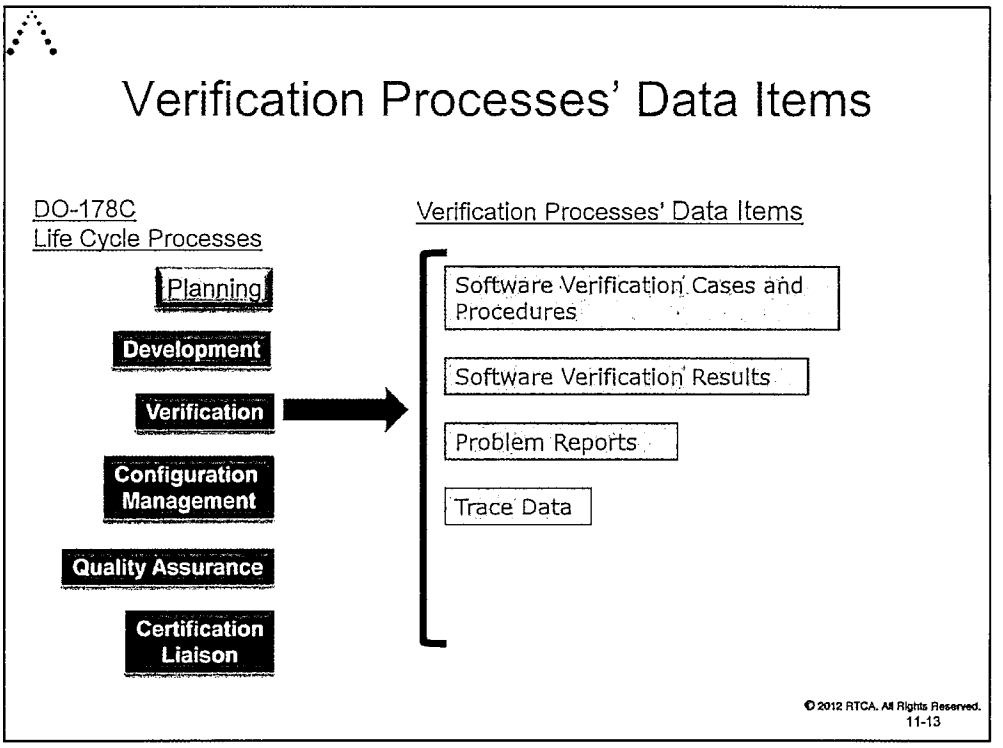


The next series of slides reviews Modules 5 – 10 Software Life Cycle Data.

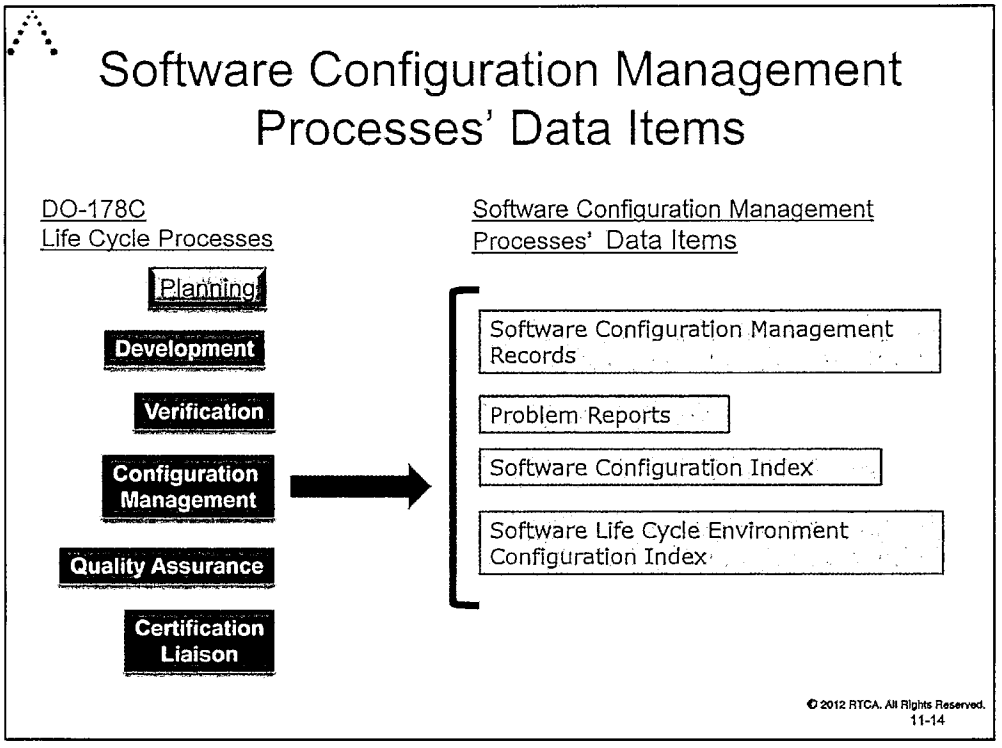
This chart was shown before in Module 5, and it is reshown to remind the participant of the data items in the Planning process.



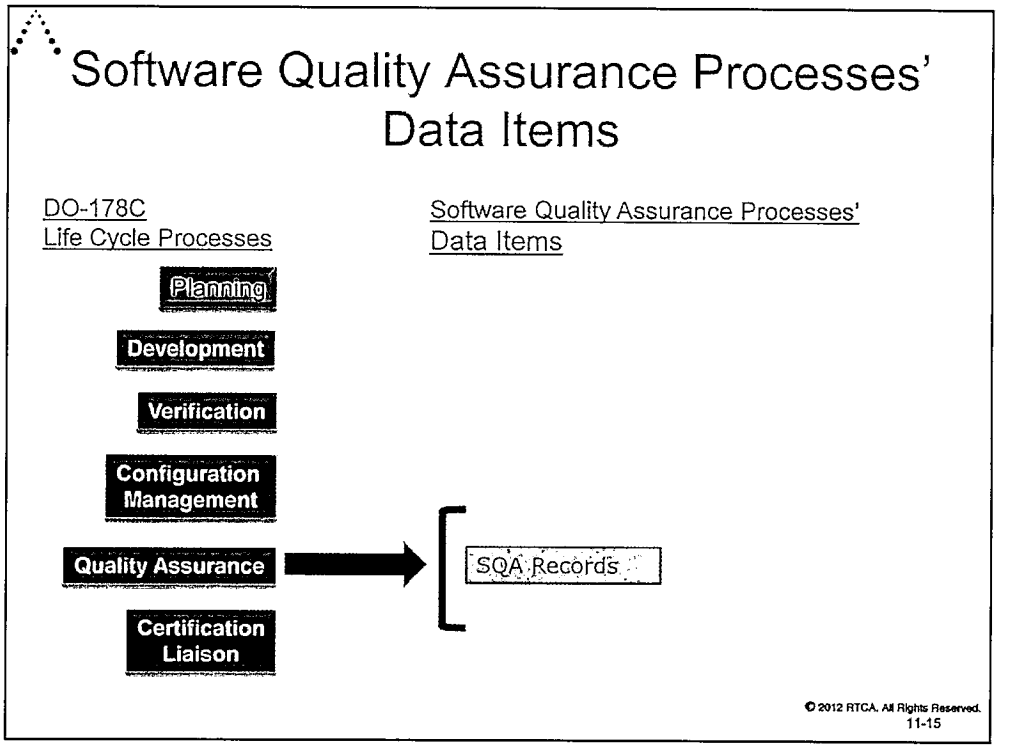
Note: Parameter Data Item File is new to DO-178C



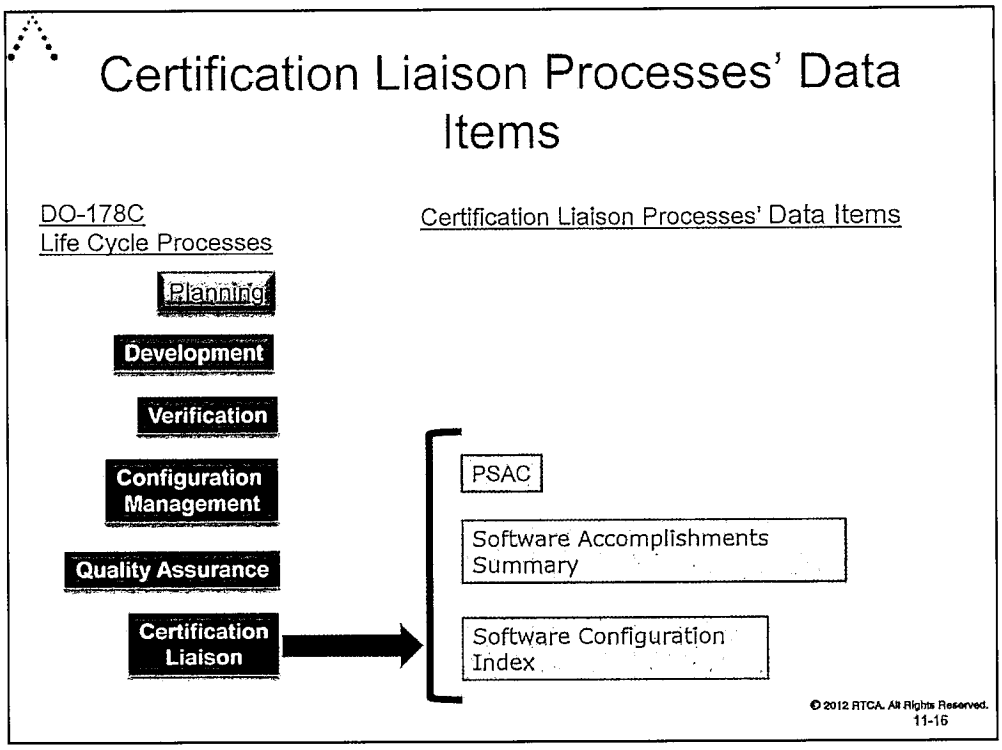
This chart was shown before in Module 7, and it is reshown to remind the participant of the data items in the Planning process.



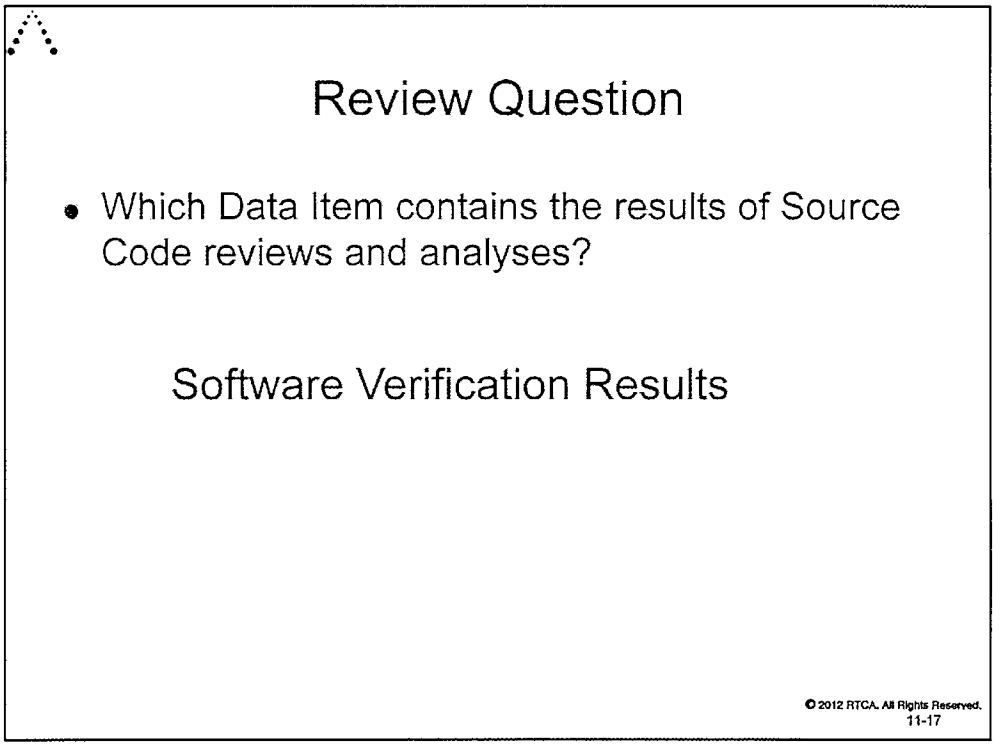
This chart was shown before in Module 8, and it is reshown to remind the participant of the data items in the Planning process.



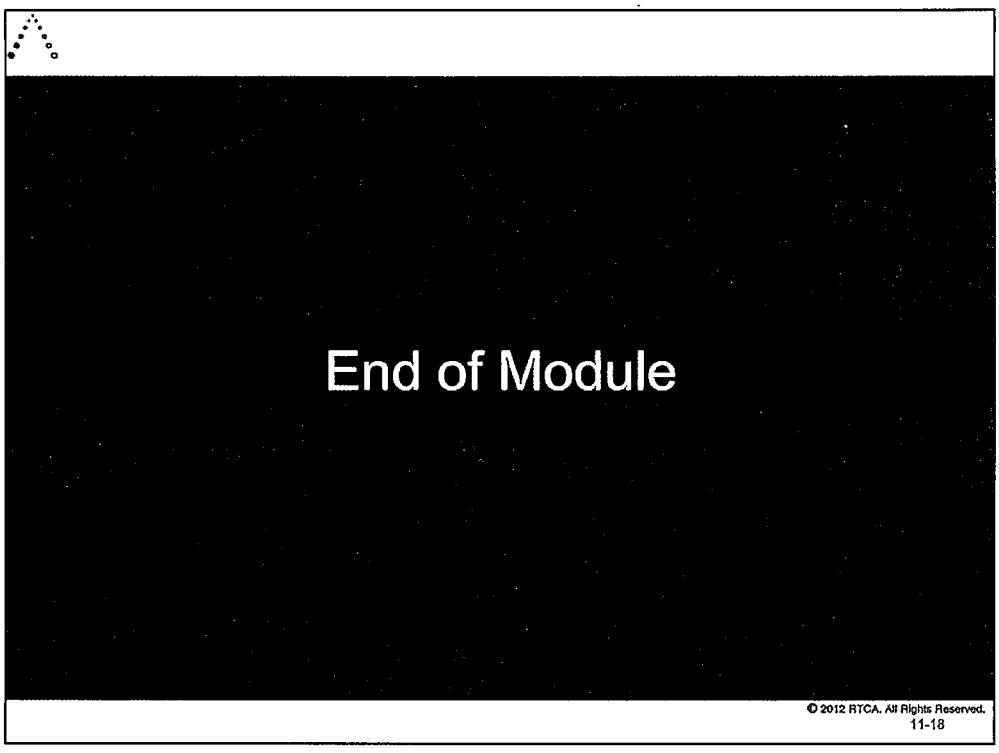
This chart was shown before in Module 9, and it is reshown to remind the participant of the data items in the Planning process.



This chart was shown before in Module 10, and it is reshown to remind the participant of the data items in the Planning process.



This is a trick question. The answer is the catch all data item for all verification results.



11-18

BACKUP

INFORMATION ON DATA ITEMS

USED IN PREVIOUS MODULES

11-19

Plan for Software Aspects of Certification (PSAC)

- Certification Authority uses PSAC as a primary means to determine whether an applicant's proposed software life cycle is commensurate with the rigor required for level of software being developed
- PSAC includes:
  - *System* Overview
    - HW/SW functional description and allocation, interfaces, architecture and safety features
  - Software Overview
    - Software functions emphasizing safety and partitioning concepts

Plan for Software Aspects of Certification (PSAC) (Concluded)

- Certification Considerations
  - Includes certification basis, means for compliance and potential software contributions to failure conditions
- Software Life Cycle
  - Involves each life cycle process objectives to be met, organizations and their responsibilities
- Software Life Cycle Data
  - Data produced and controlled by each software life cycle process, data relationships and data to be submitted to Certification Authority
- Schedule
- Additional considerations
  - Alternative methods for software compliance
- Supplier Oversight
  - Ensuring supplier's processes and outputs comply with approved software plans and standards

Software Development Plan (SDP)

- Describe software development procedures and software life cycle(s) to satisfy software development process objectives. It includes:
  - Software Standards for Requirements, Design, Code
  - Description of the software life cycle processes to be used, including the transition criteria, distinct from the PSAC by providing necessary detail
  - Software development environment
    - Requirements development, Design and Coding methods and hardware platforms for tools to be used

Software Verification Plan (SVP)

- Describes procedures to satisfy software verification objectives including
  - Organizational responsibilities
  - Interfaces with Software Life Cycle Processes
  - Independent verification methods (Review, Analysis and Testing) for each verification process activity
  - Equipment description and application of Testing and Analysis tools
  - Transition criteria
  - Partitioning considerations and their integrity
  - Compiler/linkage editor correctness
  - Methods for identifying, analyzing and verifying affected software areas and changed parts of Executable Object Code
  - Description of verification process activities if Multiple Version Dissimilar Software is used

Software Configuration Management (SCM) Plan

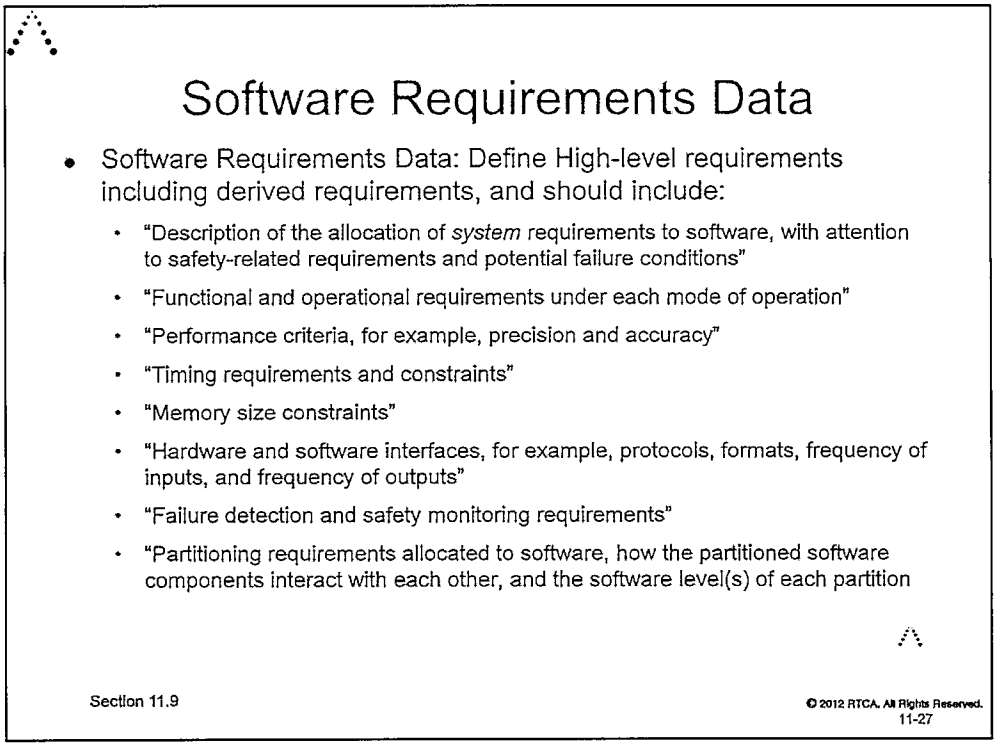
- Establishes methods to achieve SCM process objectives including
  - Description of procedures, tools, standards, organizational responsibilities and interfaces to be used
  - Description of activities such as Configuration Identification, Baseline and Traceability, Problem Reporting, Change Control, Change Review, Configuration Status Accounting, Archive, Retrieval and Release, Software Load Control, Life Cycle Environment and Data Controls
  - Determination of Control Category (CC1 and CC2) for each Software Life Cycle Data item

Software Quality Assurance (SQA) Plan

- Establishes methods for achieving SQA process objectives including descriptions of:
  - Process improvements, metrics and progressive management methods
  - SQA environment
  - SQA authority, responsibility and independent software products approval
  - SQA activities throughout the life cycle related to problem reporting, tracking and corrective action *system*
  - Transition criteria
  - SQA process activities timing
  - SQA records
  - Supplier oversight

Software Standards

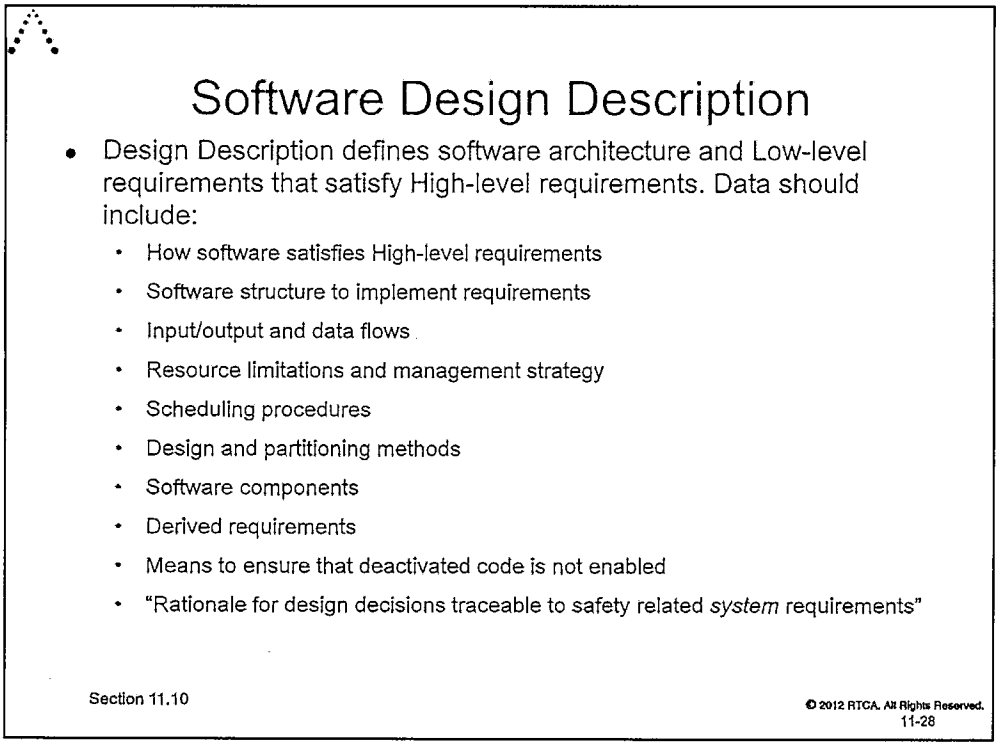
- **Requirements Standards:** Define methods, rules and tools to be used to develop High-level requirements including
  - Structured methods, notations expressing requirements, constraints on tool use and method to provide derived requirements to *system* processes
- **Design Standards:** Define methods, rules and tools to be used to develop software architecture and Low-level requirements including
  - Design description method(s), naming convention, conditions imposed on permitted design methods, constraints on design and related tool use and complexity restrictions
- **Code Standards:** Define programming languages, methods, rules, and tools to be used to code software including
  - Source code presentation standards, naming conventions and associated constraints and conditions
- **Software Model Standards** (DO-331 - Model-Based Development supplement): define modeling techniques for each type of model including
  - Methods and tools, modeling languages, style guidelines and complexity restrictions, constraints on use, requirements identification methods, model element identification, and rational for suitability



Keep the requirements positive. DO NOT have a requirement that uses the phrase "shall not". For purposes of DO-178C, requirements should specify functionality.

Trace data will be discussed in the last "Software Development Process Traceability" charts at the end of this module. The information here is included for completeness.

The trace data specified in this slide may be created during the sub-process that creates the Design Description or it may be later in a separate sub-process.

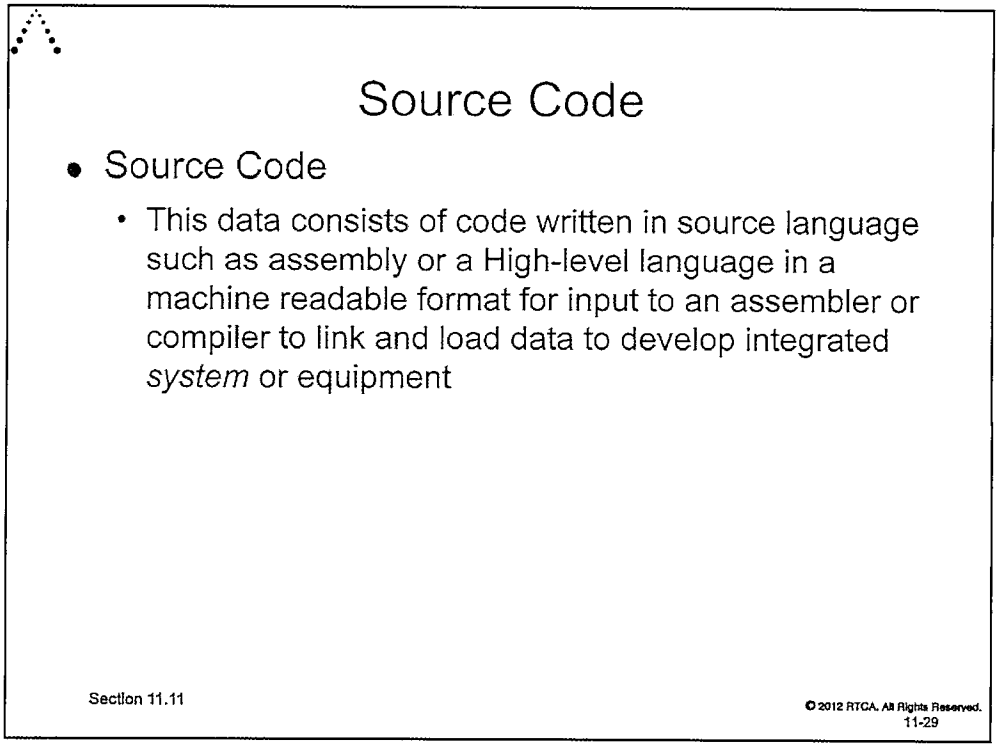


Software architecture is the structure of the software elements (for example tasks, procedures, functions, etc...) including inputs/outputs and calling /invocation methods for each software element.

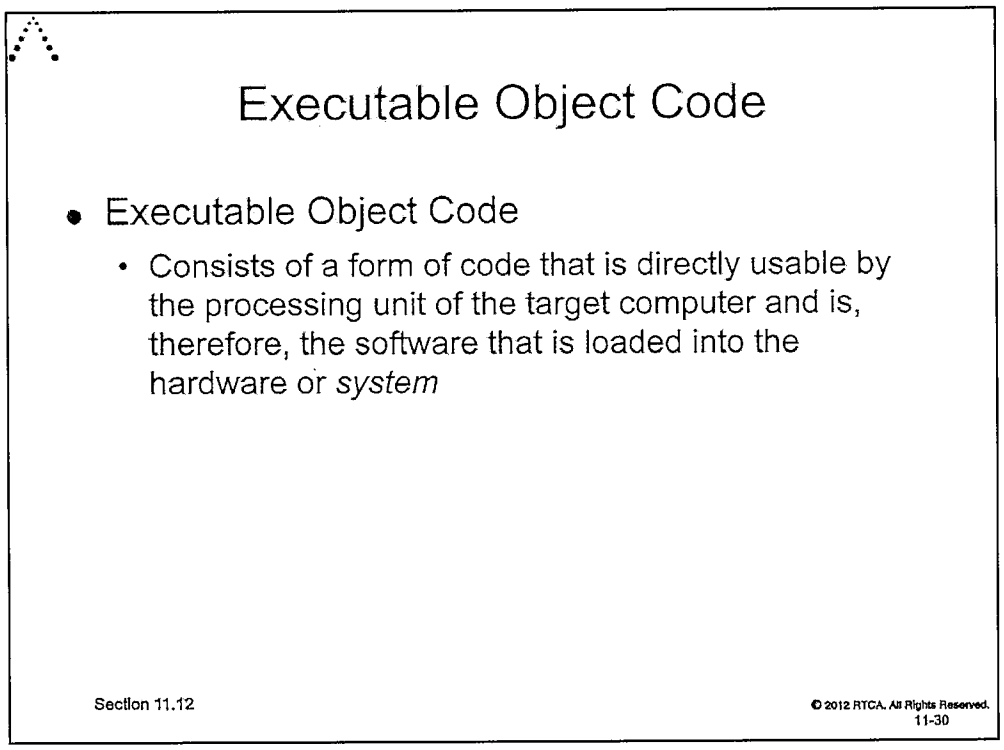
Low-level requirements specify the behavior of each software element or small group of elements. This is not to imply that there must be a Low-level requirements for each element nor does it imply that a software element might not implement more than one Low-level requirement. Keep the requirements positive. DO NOT have a requirement that uses the phrase "shall not". For purposes of DO-178C, requirements should specify functionality.

Trace data will be discussed in the last "Software Development Process Traceability" charts at the end of this module. The information here is included for completeness.

The trace data specified in this slide may be created during the sub-process that creates the Design Description or it may be later in a separate sub-process.



"The Source Code is used with the compiling, linking, and loading data in the integration process to develop the integrated *system* or equipment." The compiling, linking, and loading data maybe created in the coding sub-process or in the integration sub-process.



Executable Object Code is the implementation of the *system*/software requirements that must be tested during the verification process (section 6.4). It ends up on the airplane.

Parameter Data Item File is verified as specified in DO-178C section 6.6. It ends up on the airplane associated with the Executable Object Code.

Software Verification Cases and Procedures

- Provide details on how the software process activities are implemented and should include:
  - Review and Analysis Procedures
  - Test Cases
  - Test Procedures

Software Verification Results

- Results produced by Software Verification Process should:
  - For each review, analysis, and test, indicate each procedure that passed or failed during the activities and the final pass/fail results
  - Identify the configuration item or software version reviewed, analyzed, or tested
  - Include the results of tests, reviews, and analyses, including coverage and traceability analyses
  - Any discrepancies found should be recorded and tracked via problem reporting

Software Life Cycle Environment Configuration Index

- Identifies software life cycle environment configuration
- Aids reproduction of hardware/software life cycle environment for regeneration, reverification or modification

Software Configuration Index (SCI)

- SCI provides specific configuration and version identifiers and identifies:
  - Software product including previously developed software
  - Executable Object Code (with Data Integrity Checks) and Parameter Data Items Files with instructions to build them
  - Each source Code Component
  - Software Life Cycle Data
  - Archive and Release Media
  - Procedures, Methods and Tools for making modifications to user-modifiable software
  - Procedures and methods for loading software into target hardware

Problem Reports

- Identify and record the resolution of software product anomalous behavior, process non-compliance with plans and standards and deficiencies in Software Life cycle Data
- Reports include:
  - Identification of configuration item or/and life cycle process activity with a problem
  - Identification of configuration item to be modified or description of process to be changed
  - Detailed problem description for clear understanding, resolution with corrective action

SCM Records

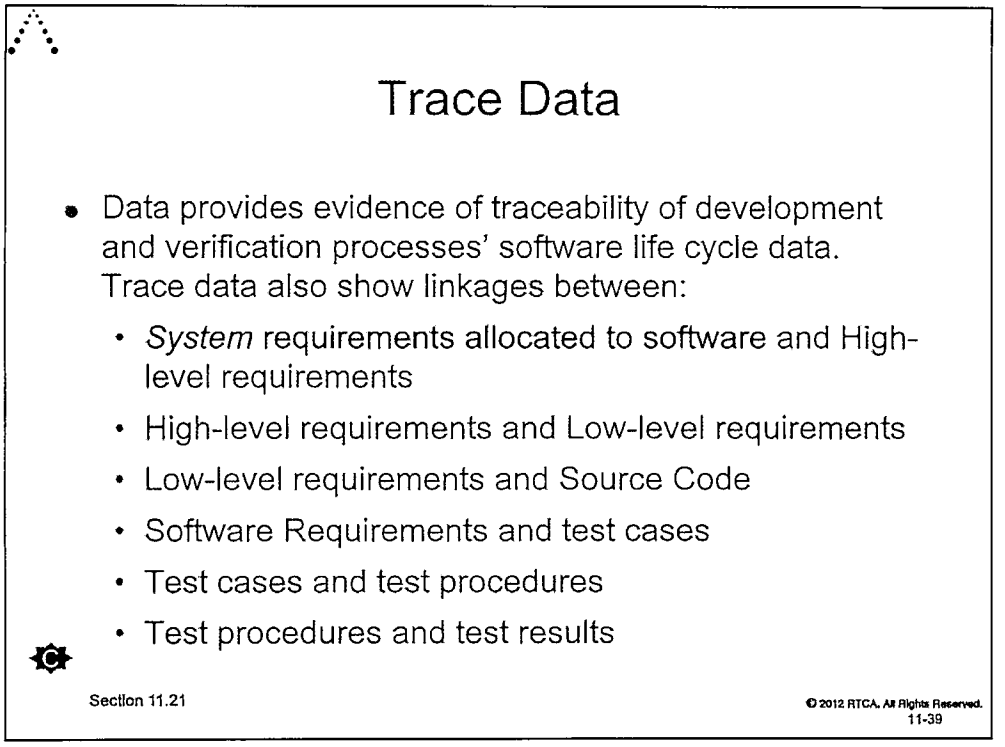
- Provide records of SCM activities, such as:
  - Configuration identification lists
  - Software library records
  - Change history records
  - Archive records
  - Release records

SQA Records

- Recorded information on SQA process activities, including:
  - SQA audits reports
  - Authorized process deviations
  - Software conformity reviews

Software Accomplishment Summary (SAS)

- SAS shows compliance with PSAC and includes:
  - *System* Overview
  - Software Overview
  - Software Identification
  - Software Characteristics
  - Software Change history
  - Software Status
  - Certification Considerations
  - Software Life Cycle and Data produced
  - Compliance Statement
  - Additional Considerations
  - Supplier Oversight



This Data Item in section 11 is new to DO-178C. However the need for trace data existed in DO-178B so this fixes a problem in DO-178B

Parameter Data Item

- Parameter Data Item
  - It is a set of data when in the form of a Data Item File to influence the behavior of the software without modifying the Executable Object code that is managed as a separate configuration item such as
    - Databases
    - Configuration Tables
- Parameter Data Item File
  - It is a representation of Parameter Data Item that is directly usable by processing unit of target computer

Section 11.22

Both These Data Items are new to DO-178C