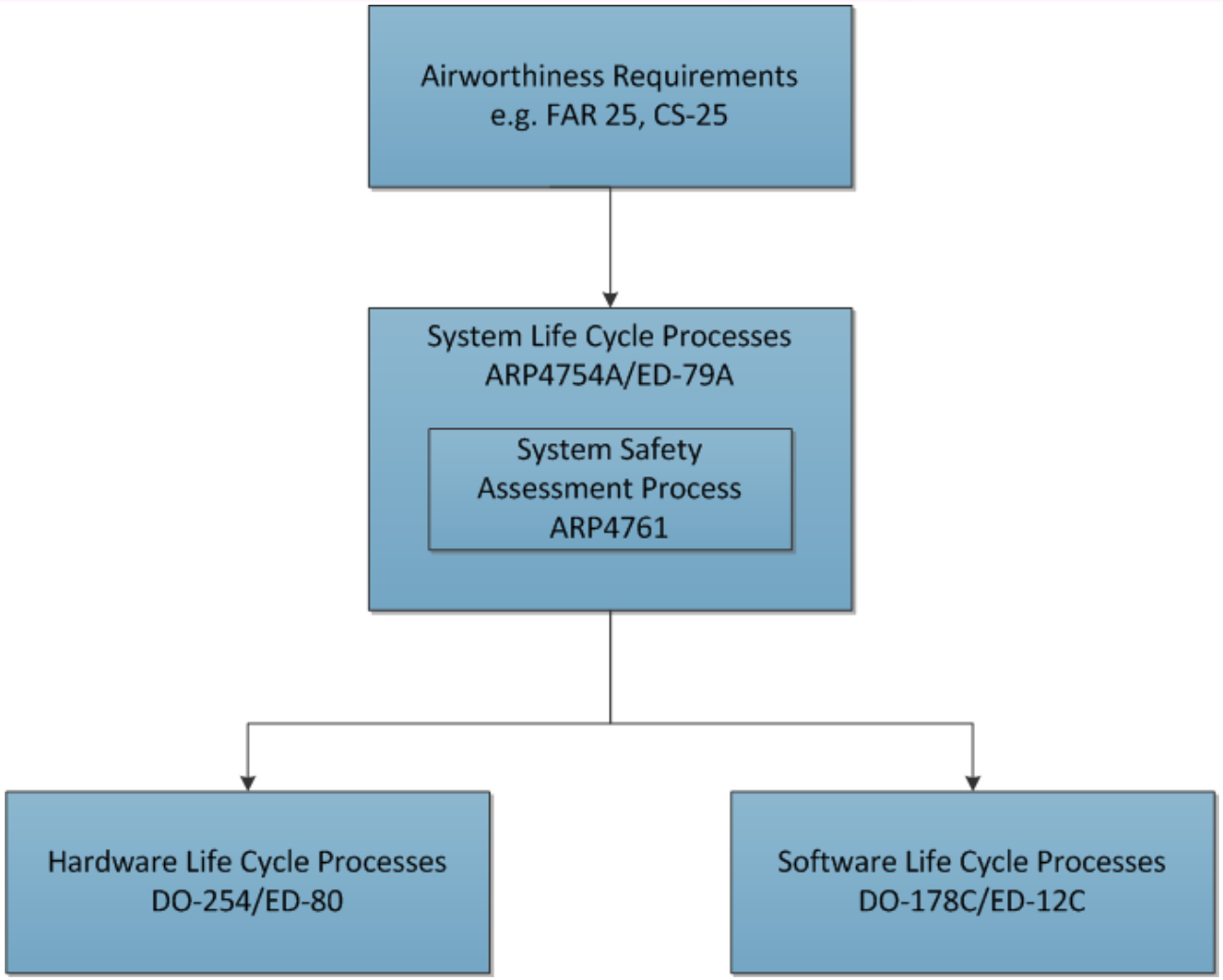
VEROCEL

The Verification Company

Software Development and Verification compliance to DO-178C/ED-12C

DO-178C/ED-12C in Context



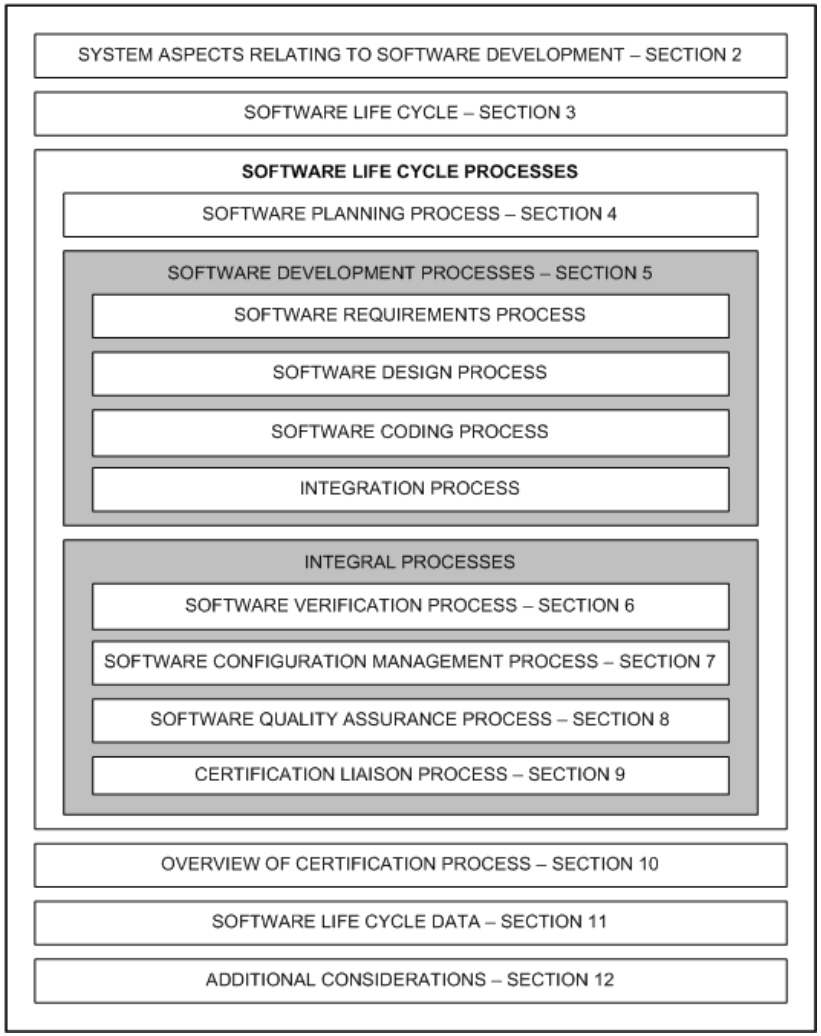
Airworthiness Requirements

- Federal Aviation Regulation (FAR) 25 — Airworthiness Standards: Transport Category Airplanes
- Certification Specification CS-25 is the European equivalent
- Others exist for gliders (CS-22), light aircraft (FAR 23/CS-23), helicopters (FAR 27/CS-27 & FAR 29/CS-29) and hot air balloons (FAR 31/CS-31HB)

CAST

- Certification Authorities Software Team
- International group of certification authority representatives
- Harmonization of certification positions on software & electronic hardware
- CAST position papers
- http://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/

Document Overview



Software Level

- Software levels determined by system safety assessment process (usually done in accordance with SAE ARP4754)
- Based on potential failure conditions
- 5 levels from Level A (the most rigorous) to Level E (the least rigorous)
- Objectives & independence varied by software level
- We'll outline these objectives in this presentation

Failure Condition

- Software criticality levels

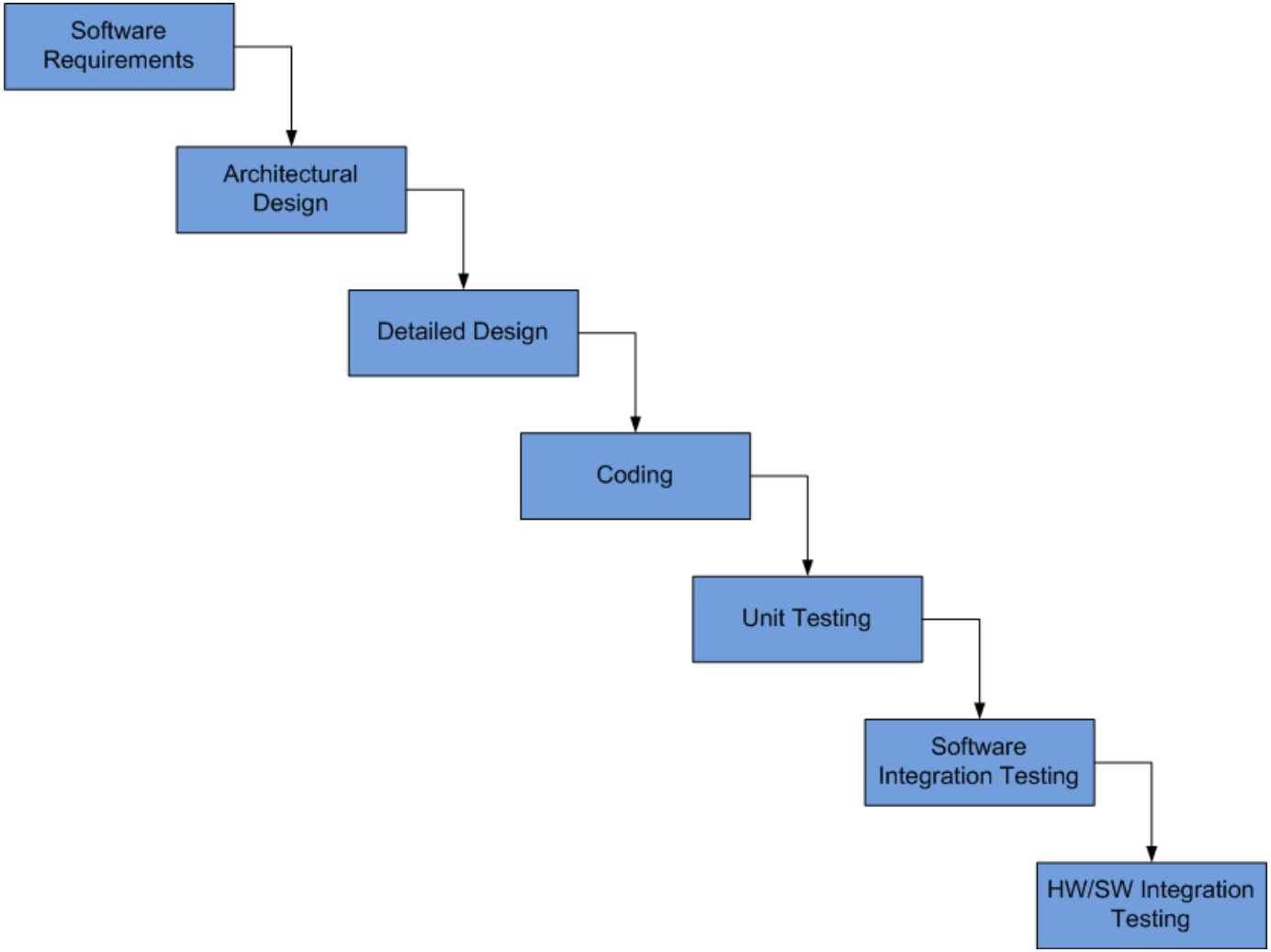
Failure Condition	Software Level
Catastrophic	Level A
Hazardous/Sever - Major	Level B
Major	Level C
Minor	Level D
No Effect	Level E

SOFTWARE LIFE-CYCLE

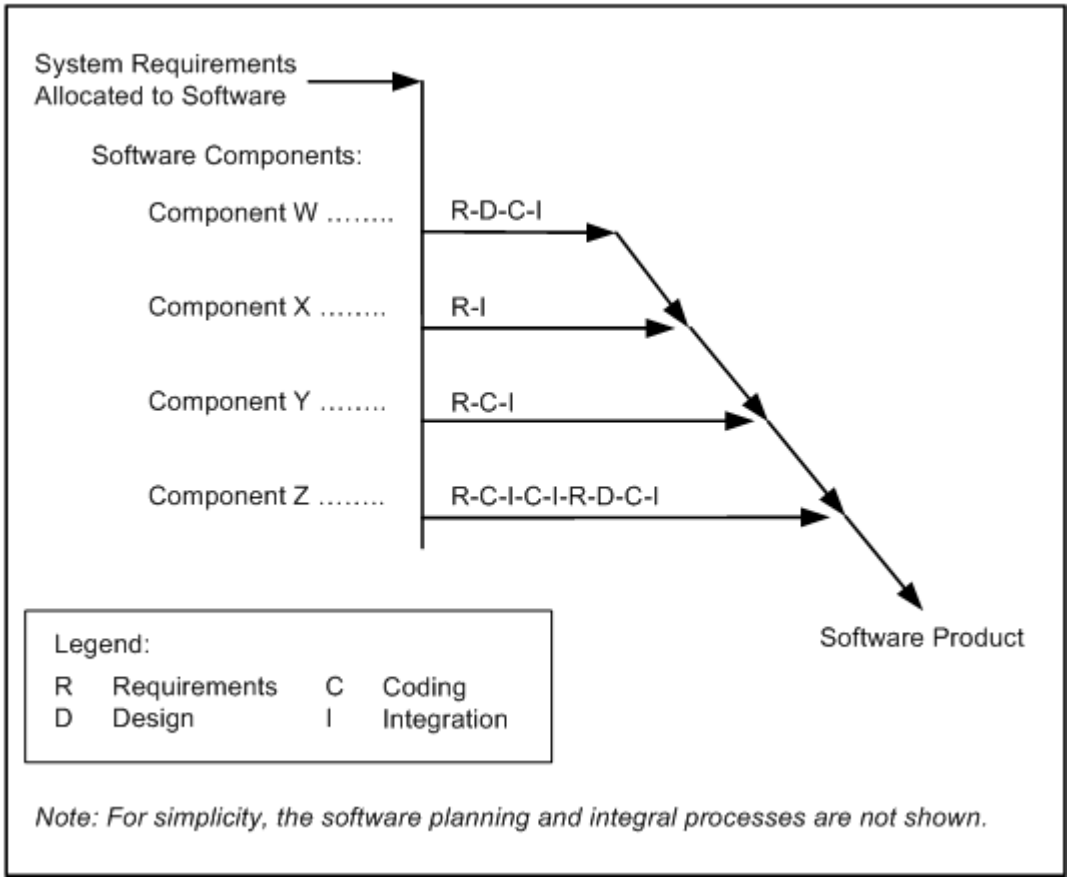
Software Life Cycle Processes

- Software planning process (DO-178C/ED-12C §4)
- Software development processes (DO-178C/ED-12C §5)
- Integral processes
  - Software verification process (DO-178C/ED-12C §6)
  - Software configuration management process (DO-178C/ED-12C §7)
  - Software quality assurance process (DO-178C/ED-12C §8)
  - Certification liaison process (DO-178C/ED-12C §9)

Conventional Waterfall Model



Example From DO-178C/ED-12C



DO-178C PROCESSES AND ACTIVITIES

Planning process

- Purpose
  - Defines the means of producing software which satisfy the system requirements and provide the level of confidence which is consistent with the airworthiness requirements
- Output:
  - Plan for Software Aspect of Certification (PSAC)
  - Software Development Plan (SDP)
  - Software Verification plan (SVP)
  - Software Quality Assurance Plan (SQPP and SQAP)
  - Software Configuration Management Plan (SCMP)
  - Design standards (SDS)

Planning process – Table A-1

Table A-1 Software Planning Process

	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	The activities of the software life cycle processes are defined.	4.1.a	4.2.a 4.2.c 4.2.d 4.2.e 4.2.g 4.2.i 4.2.l 4.3.c	O	O	O	O	PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	① ① ① ① ①	① ① ① ① ①	① ② ② ② ②	① ② ② ② ②
2	The software life cycle(s), including the inter-relationships between the processes, their sequencing, feedback mechanisms, and transition criteria, is defined.	4.1.b	4.2i 4.3.b	O	O	O		PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	① ① ① ① ①	① ① ① ① ①	① ② ② ② ②	
3	Software life cycle environment is selected and defined.	4.1.c	4.4.1 4.4.2.a 4.4.2.b 4.4.2.c 4.4.3	O	O	O		PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	① ① ① ① ①	① ① ① ① ①	① ② ② ② ②	
4	Additional considerations are addressed.	4.1.d	4.2.f 4.2.h 4.2.i 4.2.j 4.2.k	O	O	O	O	PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	① ① ① ① ①	① ① ① ① ①	① ② ② ② ②	① ② ② ② ②
5	Software development standards are defined.	4.1.e	4.2.b 4.2.g 4.5	O	O	O		SW Requirements Standards SW Design Standards SW Code Standards	11.6 11.7 11.8	① ① ①	① ① ①	② ② ②	
6	Software plans comply with this document.	4.1.f	4.3.a 4.6	O	O	O		Software Verification Results	11.14	②	②	②	
7	Development and revision of software plans are coordinated.	4.1.g	4.2.g 4.6	O	O	O		Software Verification Results	11.14	②	②	②	

Development process

- Purpose:
  - Develop the system requirements in one or more level of software requirements
  - Develop the software architecture
  - Produce the source code
  - Integrate the software components to produce executable
- Outputs
  - Software Requirement Specification (SRS)
  - Software Design Description (SDD)
  - Source Code
  - Executable object code

Development process – Table A-2

Table A-2 Software Development 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
1	High-level requirements are developed.	5.1.1.a	5.1.2.a 5.1.2.b 5.1.2.c 5.1.2.d 5.1.2.e 5.1.2.f 5.1.2.g 5.1.2.j 5.5.a	O	O	O	O	Software Requirements Data Trace Data	11.9 11.21	①	①	①	①
2	Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process.	5.1.1.b	5.1.2.h 5.1.2.i	O	O	O	O	Software Requirements Data	11.9	①	①	①	①
3	Software architecture is developed.	5.2.1.a	5.2.2.a 5.2.2.d	O	O	O	O	Design Description	11.10	①	①	①	②
4	Low-level requirements are developed.	5.2.1.a	5.2.2.a 5.2.2.e 5.2.2.f 5.2.2.g 5.2.3.a 5.2.3.b 5.2.4.a 5.2.4.b 5.2.4.c 5.5.b	O	O	O		Design Description Trace Data	11.10 11.21	① ①	① ①	① ①	
5	Derived low-level requirements are defined and provided to the system processes, including the system safety assessment process.	5.2.1.b	5.2.2.b 5.2.2.c	O	O	O		Design Description	11.10	①	①	①	
6	Source Code is developed.	5.3.1.a	5.3.2.a 5.3.2.b 5.3.2.c 5.3.2.d 5.5.c	O	O	O		Source Code Trace Data	11.11 11.21	① ①	① ①	① ①	
7	Executable Object Code and Parameter Data Item Files, if any, are produced and loaded in the target computer.	5.4.1.a	5.4.2.a 5.4.2.b 5.4.2.c 5.4.2.d 5.4.2.e 5.4.2.f	O	O	O	O	Executable Object Code Parameter Data Item File	11.12 11.22	① ①	① ①	① ①	① ①

High-Level Requirements

- Compliance with system requirements
- Accuracy and consistency
- Compatibility with the target computer
- Verifiability
- Conformance to standards
- Traceability
- Algorithm aspects

Verification of S/W requirements – Table A-3

Table A-3 Verification of Outputs of Software Requirements Process

	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	High-level requirements comply with system requirements.	6.3.1.a	6.3.1	●	●	○	○	Software Verification Results	11.14	②	②	②	②
2	High-level requirements are accurate and consistent.	6.3.1.b	6.3.1	●	●	○	○	Software Verification Results	11.14	②	②	②	②
3	High-level requirements are compatible with target computer.	6.3.1.c	6.3.1	○	○			Software Verification Results	11.14	②	②		
4	High-level requirements are verifiable.	6.3.1.d	6.3.1	○	○	○		Software Verification Results	11.14	②	②	②	
5	High-level requirements conform to standards.	6.3.1.e	6.3.1	○	○	○		Software Verification Results	11.14	②	②	②	
6	High-level requirements are traceable to system requirements.	6.3.1.f	6.3.1	○	○	○	○	Software Verification Results	11.14	②	②	②	②
7	Algorithms are accurate.	6.3.1.g	6.3.1	●	●	○		Software Verification Results	11.14	②	②	②	

Verification of S/W Design

Table A-4 Verification of Outputs of Software Design Process

	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	Low-level requirements comply with high-level requirements.	6.3.2.a	6.3.2	●	●	○		Software Verification Results	11.14	②	②	②	
2	Low-level requirements are accurate and consistent.	6.3.2.b	6.3.2	●	●	○		Software Verification Results	11.14	②	②	②	
3	Low-level requirements are compatible with target computer.	6.3.2.c	6.3.2	○	○			Software Verification Results	11.14	②	②		
4	Low-level requirements are verifiable.	6.3.2.d	6.3.2	○	○			Software Verification Results	11.14	②	②		
5	Low-level requirements conform to standards.	6.3.2.e	6.3.2	○	○	○		Software Verification Results	11.14	②	②	②	
6	Low-level requirements are traceable to high-level requirements.	6.3.2.f	6.3.2	○	○	○		Software Verification Results	11.14	②	②	②	
7	Algorithms are accurate.	6.3.2.g	6.3.2	●	●	○		Software Verification Results	11.14	②	②	②	
8	Software architecture is compatible with high-level requirements.	6.3.3.a	6.3.3	●	○	○		Software Verification Results	11.14	②	②	②	
9	Software architecture is consistent.	6.3.3.b	6.3.3	●	○	○		Software Verification Results	11.14	②	②	②	
10	Software architecture is compatible with target computer.	6.3.3.c	6.3.3	○	○			Software Verification Results	11.14	②	②		
11	Software architecture is verifiable.	6.3.3.d	6.3.3	○	○			Software Verification Results	11.14	②	②		
12	Software architecture conforms to standards.	6.3.3.e	6.3.3	○	○	○		Software Verification Results	11.14	②	②	②	
13	Software partitioning integrity is confirmed.	6.3.3.f	6.3.3	●	○	○	○	Software Verification Results	11.14	②	②	②	②

Low-Level Requirements

- Compliance with high-level requirements
- Accuracy and consistency
- Compatibility with the target computer
- Verifiability
- Conformance to standards
- Traceability
- Algorithm aspects

Software Architecture

- Compatibility with the high-level requirements
- Consistency, esp. data flow and control flow
- Compatibility with the target computer
- Verifiability
- Conformance to standards
- Partitioning integrity

Software Coding Process

- Compliance with LL requirements and architecture
- Accuracy and consistency
- Verifiability
- Conformance to standards
- Traceability
- Parameter Data Items
- Integration Process is correct

Parameter Data Items

- Parameter Data Items can be developed and verified separately if certain conditions are met
  - Can be used to configure run-time environment
- The high-level requirements describe how the software uses the parameter data items
- The low-level requirements define the structure, attributes and allowable values of the parameter data items
- Verification should show that every data element has the correct value

Coding and Integration Process – Table A-5

Table A-5 Verification of Outputs of Software Coding & Integration 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
1	Source Code complies with low-level requirements.	6.3.4.a	6.3.4	●	●	○		Software Verification Results	11.14	②	②	②	
2	Source Code complies with software architecture.	6.3.4.b	6.3.4	●	○	○		Software Verification Results	11.14	②	②	②	
3	Source Code is verifiable.	6.3.4.c	6.3.4	○	○			Software Verification Results	11.14	②	②		
4	Source Code conforms to standards.	6.3.4.d	6.3.4	○	○	○		Software Verification Results	11.14	②	②	②	
5	Source Code is traceable to low-level requirements.	6.3.4.e	6.3.4	○	○	○		Software Verification Results	11.14	②	②	②	
6	Source Code is accurate and consistent.	6.3.4.f	6.3.4	●	○	○		Software Verification Results	11.14	②	②	②	
7	Output of software integration process is complete and correct.	6.3.5.a	6.3.5	○	○	○		Software Verification Results	11.14	②	②	②	
8	Parameter Data Item File is correct and complete	6.6.a	6.6	●	●	○	○	Software Verification Cases and Procedures Software Verification Results	11.13 11.14	① ②	① ②	② ②	② ②
9	Verification of Parameter Data Item File is achieved.	6.6.b	6.6	●	●	○		Software Verification Results	11.14	②	②	②	

Verification processes

- Purpose:
  - Verification of the software requirement process
  - Verification of software design process
  - Verification of the SW coding and integration

- Challenges:
  - The cost may represent up to 50% of the total effort.

Reviews and Analyses

- Reviews provide a qualitative assessment of correctness, e.g. an inspection of an output of a process guided by a checklist or similar aid (DO-178C/ED-12C §6.3)
- Analyses provide repeatable evidence of correctness (DO-178C/ED-12C §6.3)

Reviews and Analyses

- High-Level Requirements (DO-178C/ED-12C §6.3.1)
- Low-Level Requirements (DO-178C/ED-12C §6.3.2)
- Software Architecture (DO-178C/ED-12C §6.3.3)
- Source Code (DO-178C/ED-12C §6.3.4)
- Outputs of the Integration Process (DO-178C/ED-12C §6.3.5)
- Test Cases, Procedures and Results (DO-178C/ED-12C §6.4.5)

Outputs of the Integration Process

- Detailed examination of the linking and loading data and memory map
- Topics include:
  - Incorrect hardware addresses
  - Memory overlaps
  - Missing software components

SOFTWARE TESTING AND VERIFICATION

Test Environment

- Preferred test environment includes the software loaded into the target computer and tested in a high fidelity simulation of the target computer environment
- Some testing may need to be performed on a small software component that is functionally isolated from other software components
- Selected tests should always be performed in the integrated target computer environment
- Emulators and simulators
- Tool qualification

Normal Range Test Cases

- Real and integer input variables
- Time-related functions
- State transitions
- Software requirements expressed by logic equations

Equivalence Classes

- Exhaustive testing is impractical for non-trivial programs
- Equivalence class: “The partition of the input domain of a program such that a test of a representative value of the class is equivalent to a test of other values of the class” (DO-178C/ED-12C Glossary)

Robustness Testing

- Real and integer variables
- System initialization during abnormal conditions
- Possible failure modes of the incoming data
- Loops
- Protection mechanisms for exceeding frame times
- Time-related functions
- State transitions

Testing of Integration Process – 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
1	Executable Object Code complies with high-level requirements.	6.4.a	6.4.2 6.4.2.1 6.4.3 6.5	O	O	O	O	Software Verification Cases and Procedures Software Verification Results Trace Data	11.13 11.14 11.21	① ② ①	① ② ①	② ② ②	② ② ②
2	Executable Object Code is robust with high-level requirements.	6.4.b	6.4.2 6.4.2.2 6.4.3 6.5	O	O	O	O	Software Verification Cases and Procedures Software Verification Results Trace Data	11.13 11.14 11.21	① ② ①	① ② ①	② ② ②	② ② ②
3	Executable Object Code complies with low-level requirements.	6.4.c	6.4.2 6.4.2.1 6.4.3 6.5	●	●	O		Software Verification Cases and Procedures Software Verification Results Trace Data	11.13 11.14 11.21	① ② ①	① ② ①	② ② ②	
4	Executable Object Code is robust with low-level requirements.	6.4.d	6.4.2 6.4.2.2 6.4.3 6.5	●	O	O		Software Verification Cases and Procedures Software Verification Results Trace Data	11.13 11.14 11.21	① ② ①	① ② ①	② ② ②	
5	Executable Object Code is compatible with target computer.	6.4.e	6.4.1.a 6.4.3.a	O	O	O	O	Software Verification Cases and Procedures Software Verification Results	11.13 11.14	① ②	① ②	② ②	② ②

Verification of Verification Process – Table A-7

	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	Test procedures are correct.	6.4.5.b	6.4.5	●	○	○		Software Verification Results	11.14	②	②	②	
2	Test results are correct and discrepancies explained.	6.4.5.c	6.4.5	●	○	○		Software Verification Results	11.14	②	②	②	
3	Test coverage of high-level requirements is achieved.	6.4.4.a	6.4.4.1	●	○	○	○	Software Verification Results	11.14	②	②	②	②
4	Test coverage of low-level requirements is achieved.	6.4.4.b	6.4.4.1	●	○	○		Software Verification Results	11.14	②	②	②	
5	Test coverage of software structure (modified condition/decision coverage) is achieved.	6.4.4.c	6.4.4.2.a 6.4.4.2.b 6.4.4.2.d 6.4.4.3	●				Software Verification Results	11.14	②			
6	Test coverage of software structure (decision coverage) is achieved.	6.4.4.c	6.4.4.2.a 6.4.4.2.b 6.4.4.2.d 6.4.4.3	●	●			Software Verification Results	11.14	②	②		
7	Test coverage of software structure (statement coverage) is achieved.	6.4.4.c	6.4.4.2.a 6.4.4.2.b 6.4.4.2.d 6.4.4.3	●	●	○		Software Verification Results	11.14	②	②	②	
8	Test coverage of software structure (data coupling and control coupling) is achieved.	6.4.4.d	6.4.4.2.c 6.4.4.2.d 6.4.4.3	●	●	○		Software Verification Results	11.14	②	②	②	
9	Verification of additional code, that cannot be traced to Source Code, is achieved.	6.4.4.c	6.4.4.2.b	●				Software Verification Results	11.14	②			

Test Coverage Analysis

- Requirements-based test coverage analysis
- Structural coverage analysis

Requirements Coverage Analysis

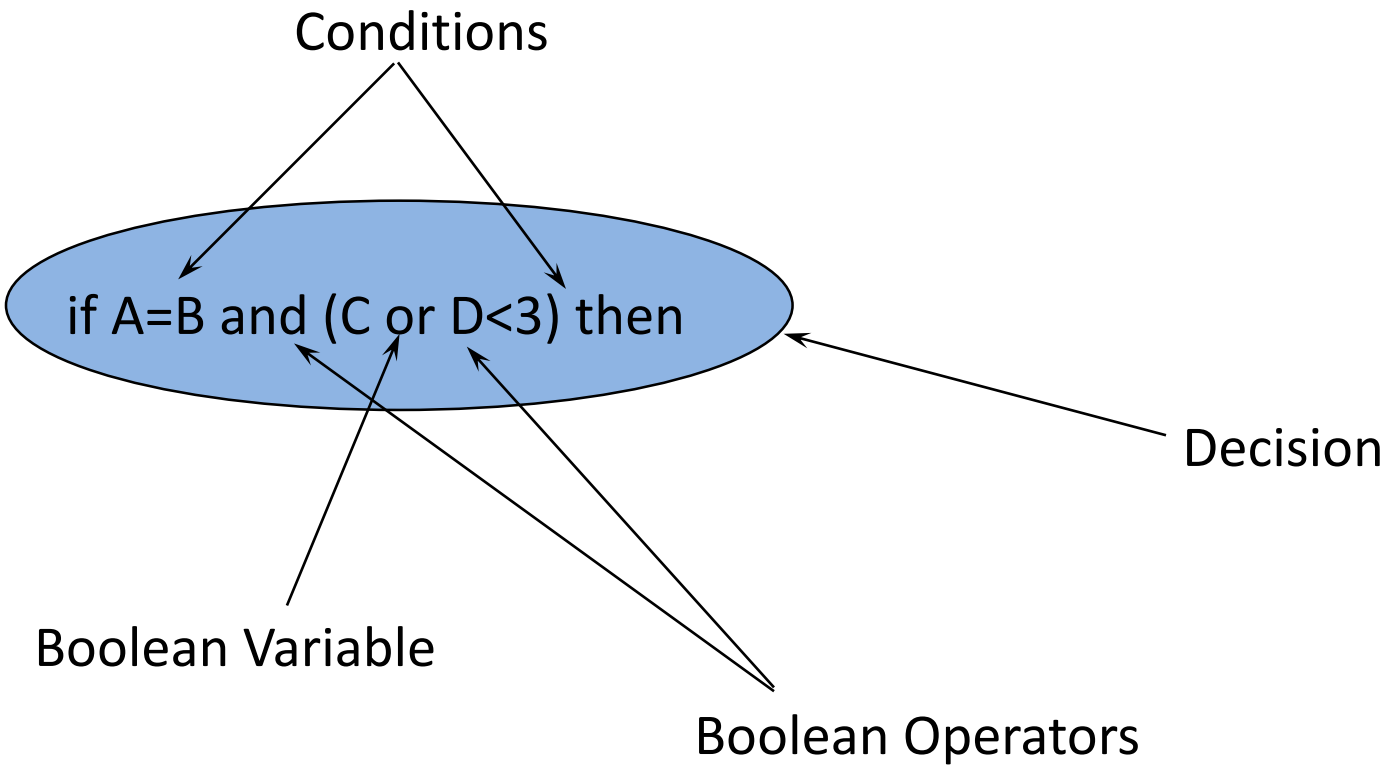
- Test cases exist for each software requirement
- Test cases satisfy the criteria of normal and robustness testing
- Test coverage of high-level requirements required at Levels A, B, C and D (with independence at Level A)
- Test coverage of low-level requirements not required at Level D

Structural Coverage Analysis

- MC/DC
- Decision Coverage
- Statement Coverage
- Data Coupling and Control Coupling
- All test cases used to achieve structural coverage should be traceable to requirements

Structural coverage

- Terminology



Decision coverage

- Boolean expressions tested in control structures (such as the if-statement and while-statement) must be evaluated to both true and false. Additionally, this measure includes coverage of switch-statement cases, exception handlers, and interrupt handlers.
- For the decision (A *or* B), test cases (*TF*) and (*FF*) will toggle the decision outcome between *true* and *false*. However, the effect of B is not tested; that is, those test cases cannot distinguish between the decision (A *or* B) and the decision A.

Condition coverage

- Requires that each condition in each decision evaluate to both TRUE and FALSE at least once
- For the decision (A *or* B) test cases (*TF*) and (*FT*) meet the coverage criterion, but do not cause the decision to take on all possible outcomes.
- As with decision coverage, a minimum of two tests cases is required for each decision.

Condition Decision coverage

- Combines the requirements for decision coverage with those for condition coverage. That is, there must be sufficient test cases to toggle the decision outcome between true and false and to toggle each condition value between true and false. Hence, a minimum of two test cases are necessary for each decision.
- Consider the following C/C++ code fragment:

```
if ( A>=0 or B>=0 )   /* supposed to be a and */
   C = sqrt (A) + sqrt (B);
```

  - Tested OK with (1 , 1) and (-1, -1). Will fail with (1,-1) and (-1,1).

MC/DC

- The MC/DC criterion enhances the condition/decision coverage criterion by requiring that each condition be shown to independently affect the outcome of the decision. The independence requirement ensures that the effect of each condition is tested relative to the other conditions.

- In general, a minimum of N+1 test cases for a decision with N inputs. For the example (A or B), test cases (TF), (FT), and (FF) provide MC/DC. For decisions with a large number of inputs, MC/DC requires considerably more test cases than any of the coverage measures discussed above.

Structural coverage

- Must account for “hidden” decision:

  A = (C and D);

  if (A)

  /* something */

  A decision is not synonymous with a branch point. MC/DC applies to all decisions, not just those within a branch point.

- And also :

  A = B or C; (statement 1)

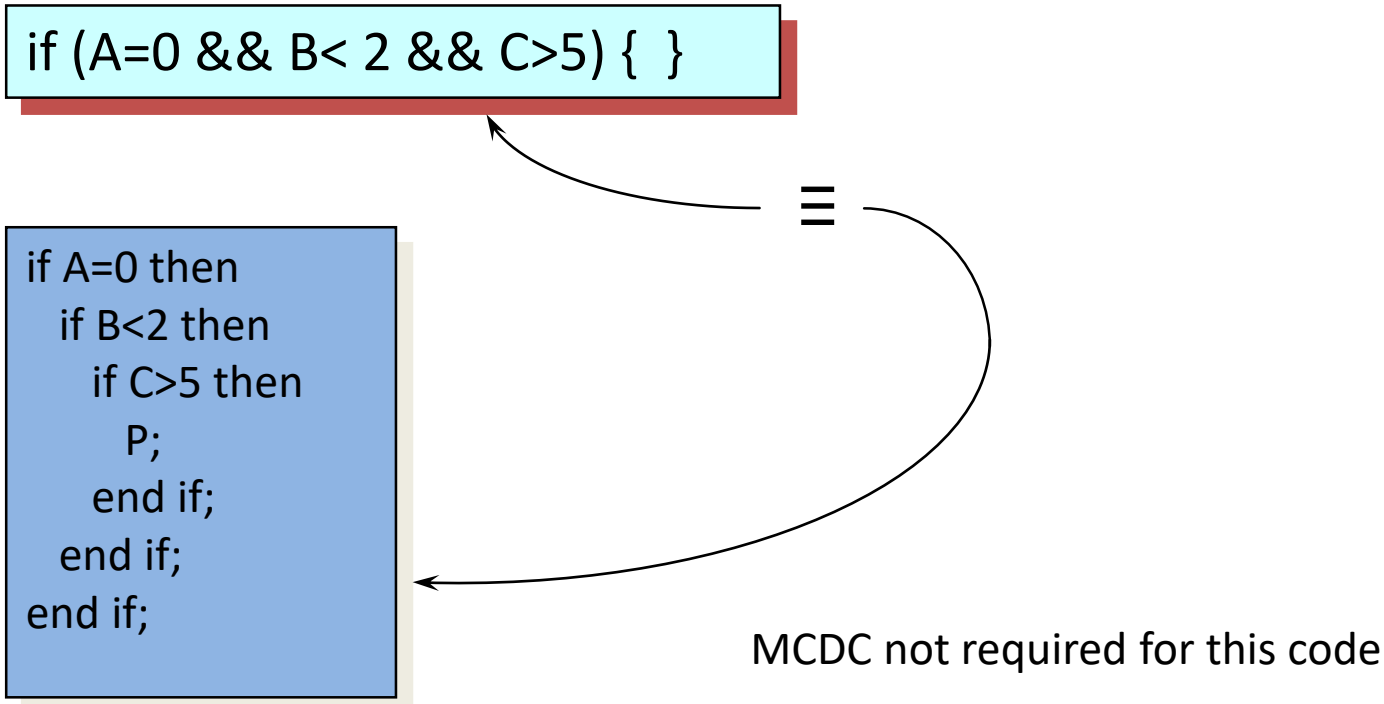
  E = A and D; (statement 2)

  These two statements are logically equivalent to:

  E = (B or C) and D; (statement 3)

- A test set that provides MC/DC for statements 1 and 2 individually will not necessarily provide MC/DC for statement 3. For this example, tests (*TFT*), (*FTF*), and (*FFT*) for (B,C,D) provide MC/DC for statements 1 and 2 individually, but do not provide MC/DC for statement 3.

Coverage at Level A



- At the object code level, MCDC is equivalent to decision coverage.

Data Coupling and Control Coupling

- Data coupling – The dependence of a software component on data not exclusively under the control of that component (DO-178C/ED-12C Glossary)
- Control coupling – The manner or degree by which one software component influences the execution of another software component (DO-178C/ED-12C Glossary)

Verification of Data & Control Coupling

- Reviews and analysis of Software Architecture (DO-178C/ED-12C §6.3.3.b)
- Reviews and analysis of Source Code (DO-178C/ED-12C §6.3.4.b)
- Requirements-based testing, confirmed by structural coverage analysis (DO-178C/ED-12C §6.4.4.d)

Analysis of Data & Control Coupling

- “Test coverage of software structure, both data coupling and control coupling, is achieved” (DO-178C/ED-12C §6.4.4.d)
- “Analysis to confirm that the requirements-based testing has exercised the data and control coupling between code components” (DO-178C/ED-12C §6.4.4.2.c)
- The intent behind this objective is to ensure that applicants do a sufficient amount of hardware/software integration testing and/or software integration testing (DO-248C/ED-94C FAQ #67)

Structural Coverage Analysis Resolution

- Shortcomings in requirements-based test cases or procedures
- Inadequacies in software requirements
- Dead code
- Deactivated code

SOFTWARE CONFIGURATION MANAGEMENT

CM process

- Purpose
  - Provide defined and controlled configuration of the software
  - Provide the ability to consistently replicate the excutable object code (or re-generate it if needed)
  - Provide consistency and repeatability in the process activities
  - Provide baselines and know points for reviews
  - Provide controls to ensure problems receive attention and changes are recorded, approved and implemented

CM process

Table A-8 Software Configuration Management Process

	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	Configuration items are identified.	7.1.a	7.2.1	O	O	O	O	SCM Records	11.18	②	②	②	②
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
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
4	Archive, retrieval, and release are established.	7.1.g	7.2.7	O	O	O	O	SCM Records	11.18	②	②	②	②
5	Software load control is established.	7.1.h	7.4	O	O	O	O	SCM Records	11.18	②	②	②	②
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

SOFTWARE QUALITY ASSURANCE

Quality Assurance Process

- Purpose
  - Provide assurance that SW development and integral process comply with the approved plans and standards
  - Provide assurance that transition criteria for processes are satisfied
  - Provide assurance that a conformity review of the software product is conducted.

QA process

Table A-9 Software Quality Assurance Process

	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

Certification Liaison

- Purpose :
  - Establish communication and understanding between the applicant and the certification

Table A-10 Certification Liaison Process

	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	Communication and understanding between the applicant and the certification authority is established.	9.a	9.1.b 9.1.c	O	O	O	O	Plan for Software Aspects of Certification	11.1	①	①	①	①
2	The means of compliance is proposed and agreement with the Plan for Software Aspects of Certification is obtained.	9.b	9.1.a 9.1.b 9.1.c	O	O	O	O	Plan for Software Aspects of Certification	11.1	①	①	①	①
3	Compliance substantiation is provided.	9.c	9.2.a 9.2.b 9.2.c	O	O	O	O	Software Accomplishment Summary Software Configuration Index	11.20 11.16	① ①	① ①	① ①	① ①

Certification Evidence (Life cycle data)

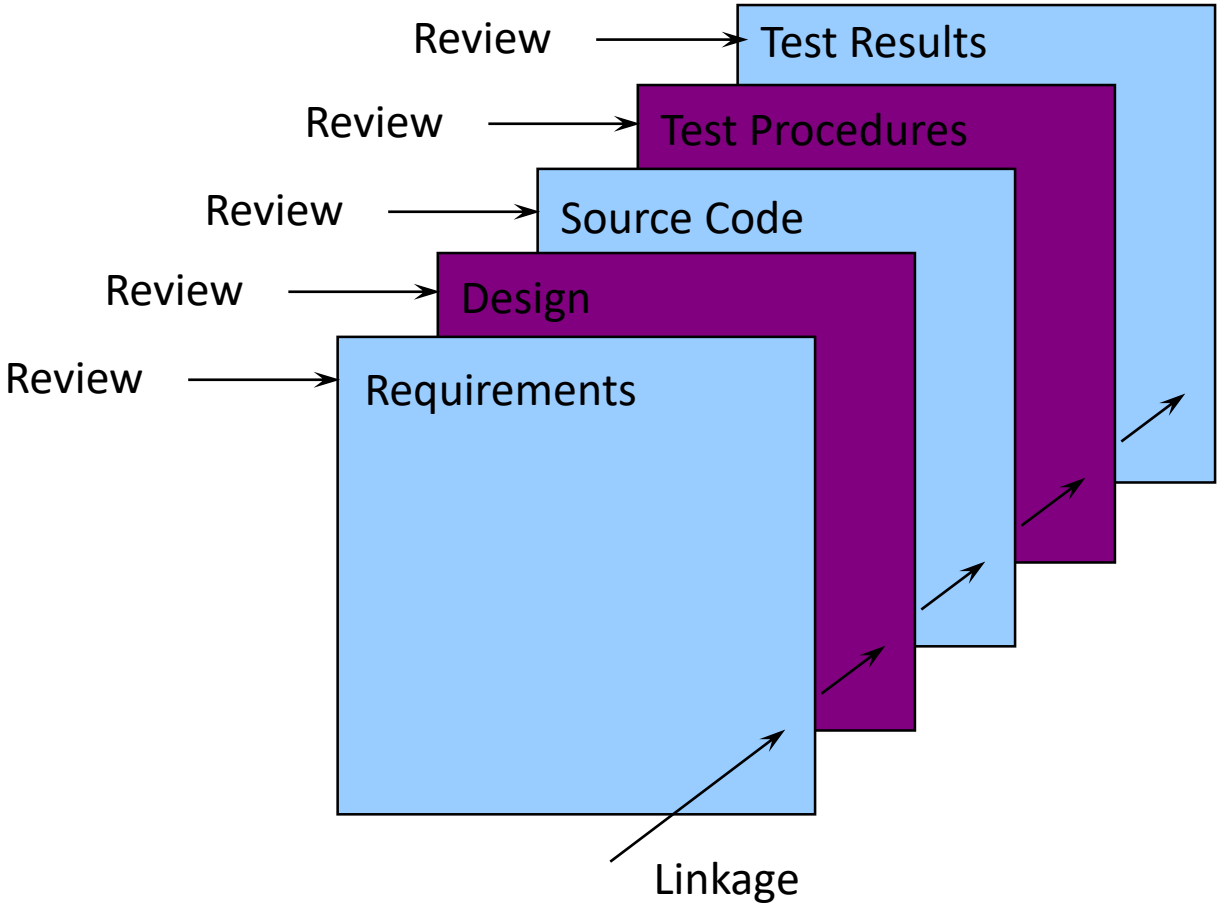
- Plan for software aspects of certification (PSAC)
- Software quality assurance plan
- Software configuration management plan
- Software development plan
  - Software requirements standards
  - Software design standards
  - Software coding standards
- Software verification plan
- Software requirements specification
- Software design document
- Version description document
- Traceability matrix
- Software development folder
  - Design reviews
  - Code reviews
  - Test reviews
  - Functional tests
  - Coverage results
- Tool qualification documentation
- Software accomplishment summary (SAS)

Software Verification Results

- Detailed and overall pass/fail results
- Configuration item or software version verified
- Results of tests, reviews and analyses

TOOLING CONSIDERATIONS

How to Prove Traceability?



Verocel VeroTrace

- *VeroTrace*
  - Verification Life-Cycle Management Tool
  - Manages Requirements, Design, Tests, Coverage, Problem Reports, and more.
  - Provides full Traceability between all of the Artifacts
    - Eases showing completeness of traceability
  - Enforces Software Development Processes
  - Impact Analysis for Changes
  - Generates Browseable Certification Evidence (on DVD)
  - Qualified to DO-330, TQL-5

Verocel Tools – Verification Tools

- *VerOCode*
  - Level A Object Code Coverage tool
  - Test on target without instrumenting the code
  - Addresses MCDC coverage
  - Qualified to DO-330, TQL-5
- *VeroSource*
  - Level A Source-based coverage tool
  - Qualified to DO-330, TQL-5
- *VeroLink*
  - Satisfies Control Coupling criteria
  - Qualified to DO-330, TQL-5
- *VeroStack*
  - Measures and calculates Worst Case stack use
  - Qualified DO-330, TQL-5
- *PICSim*
  - Instruction level simulator, Coverage Analyzer, Test Manger
  - Qualified

Tool Qualification

- Tool qualification is necessary when DO-178C/ED-12C processes are eliminated, reduced or automated by use of a software tool without its output being verified (DO-178C/ED-12C §12.2.1)
- Tool qualification is handled quite differently in DO-178C/ED-12C compared to DO-178B/ED-12B

Tool Qualification

Criteria 1: A tool whose output is part of the airborne software and thus could introduce an error

Criteria 2: A tool that is used to justify eliminating a development process or a verification process other than the one automated by the tool

Criteria 3: Any other tool that could fail to detect an error

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

The End