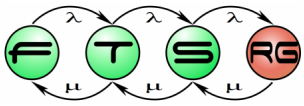
Standards in Avionics System Development
(Overview on DO-178B)

Ákos Horváth

Dept. of Measurement and Information Systems



Budapest University of Technology and Economics
Department of Measurement and Information Systems

MŰEGYETEM 1782

Abstract

- DO-178B (and DO-278) are used to assure safety of avionics software. These documents provide guidance in the areas of SW development, configuration management, verification and the interface to approval authorities (e.g., FAA, EASA)

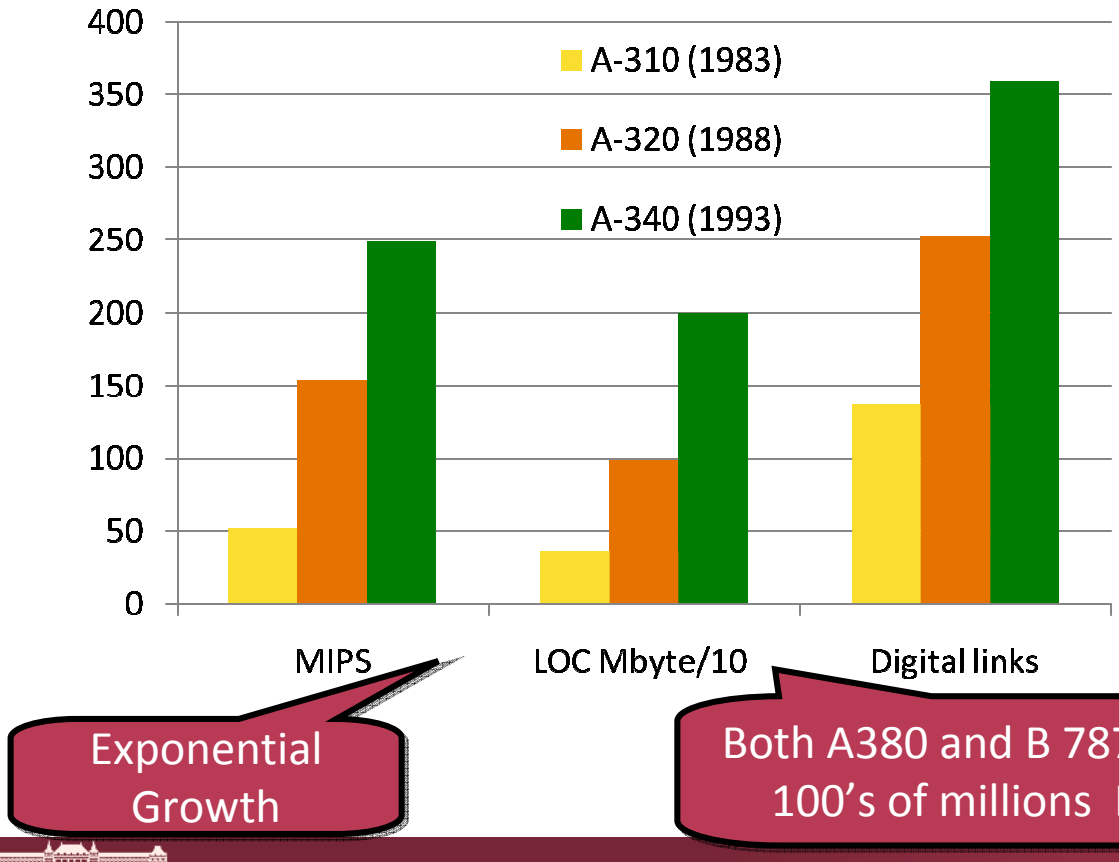
Agenda

- Introduction to DO-178B
- System Aspects
- Software Lifecycle Management
- Certification Artifacts and Techniques
- Future: DO-178C

Overview

- **DO-178B - Software Considerations in Airborne Systems and Equipment Certification**
- Standard of RTCA Incorporation (in Europe it is ED-12B and standard of EUROCAE)
- Represents the avionics industry consensus to ensure software safety
- Acceptable by FAA and EASA certification authorities
- „The FAA and the civil aviation community recognize RTCA'S DO-178B as an acceptable means of compliance to the FAA regulations for SW aspects of certification."

History of avionics SW complexity

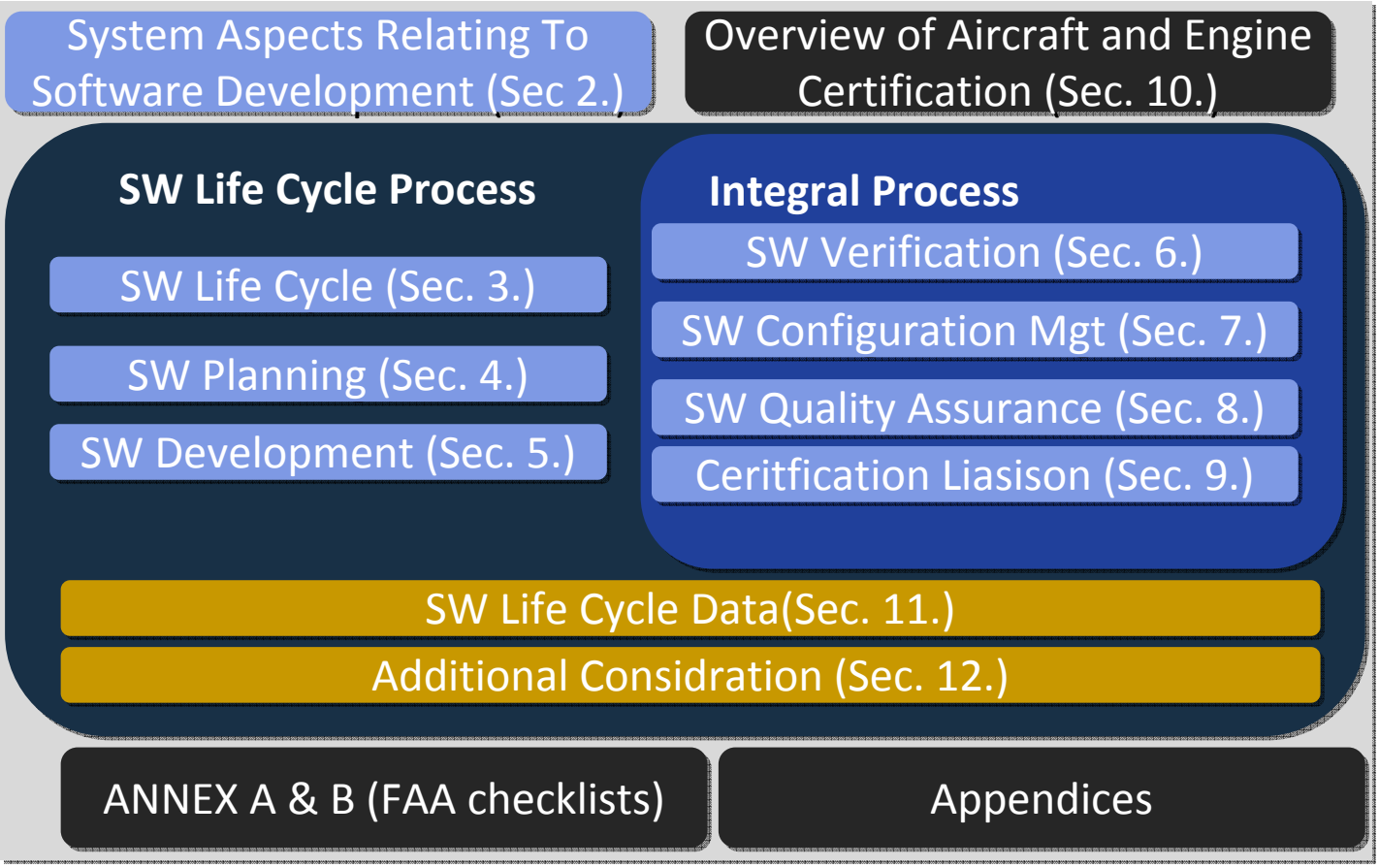


Ref: Subra de Salafa and Paquier

History

- DO-178 in 1982
  - Basic concepts of SW design assurance
  - Three levels of SW safety
- DO-178A in 1985
  - Concentrates on testing and configuration management
- DO-178B in 1992
  - Five levels of SW safety
  - From Testing focus → requirement-based
- DO-278 in 2002
  - Interprets DO-178B to ground and space based-systems
- DO-178C underway (awaited in 2011)
  - Incorporates modern SW development and analysis techniques

DO178B Document Structure



Software Levels in DO-178B

- Different failure conditions require different software conditions → 5 levels

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

Examples DO-178B Safety Levels

- **Safety-Critical Levels C&D**
  - Anti-missile defense
  - Data mining
  - Health monitoring
  - Mission planning and implementation
  - Mission simulation and training
  - Network-centric operation
  - Real-time data recording and analysis
  - Self-healing communication networks
  - Telemetry
  - Weapons targeting
- **Safety-Critical Levels A&B**
  - Fly-by-wire controls
  - Auto-pilot
  - Air-traffic Separation Control
  - Glass Cockpit Information Display
  - Radar
  - Jet Engine Control
  - IFF (friend or foe)
  - Missile guidance
  - Missile launch
  - Missile self-destruct

Objectives for Safety Levels

- Different levels of safety requires different objectives to be fulfilled
  - e.g., Level A 66, Level B 65
- Defined by 10 tables in ANNEX A
- Example: Table A-6 Objective 3.

Objective		Applicability by SW Level				Output		Control Category by SW Level			
Description	**Ref**	**A**	**B**	**C**	**D**	**Descriptions**	**Ref.**	**A**	**B**	**C**	**D**
Executable Object Code compiles with low-level requirements	6.4.2.1. 6.4.3.	●	●	○		Software Verification Cases and Procedures Software Verification Results	11.13 11.14	1 2	1 2	2 2	

Objectives for Safety Levels

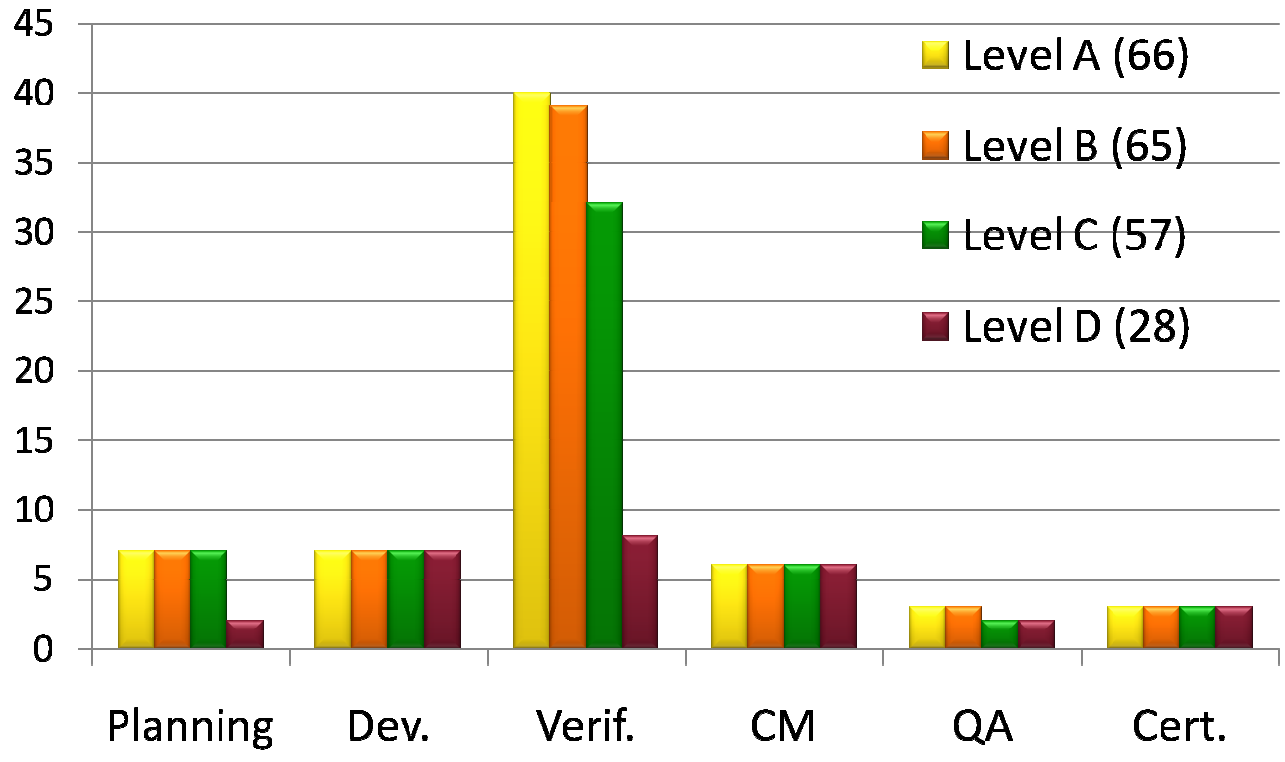
- Different levels of safety requir… to be fulfilled
  - e.g., Level A 66, Level B 65
- Defined by 10 tables in ANNEX A
- Example: Table A-6 Objective 3.

How to store the evidence

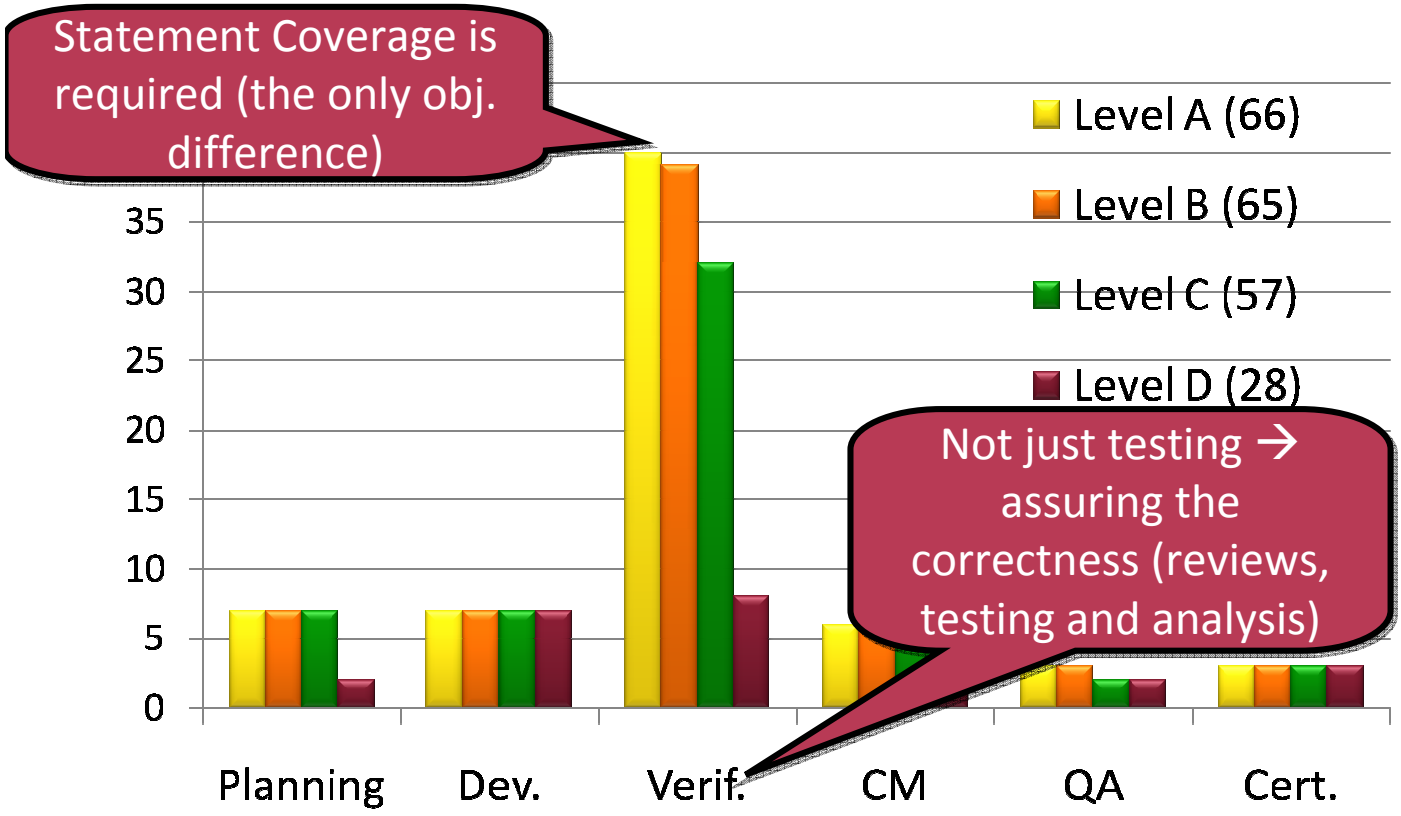
Independence is required (full means yes)

		…ity …el								
Description	**Ref**	**B**	**C**	**D**	**Output**	**Ref.**	**Control Category by SW Level**			
					Descriptions		**A**	**B**	**C**	**D**
Executable Object Code compiles with low-level requirements	6.4.2.1. 6.4.3.	●	●	○	Software Verification Cases and Procedures Software Verification Results	11.13 11.14	1 2	1 2	2 2	

Objectives Distribution in DO-178B



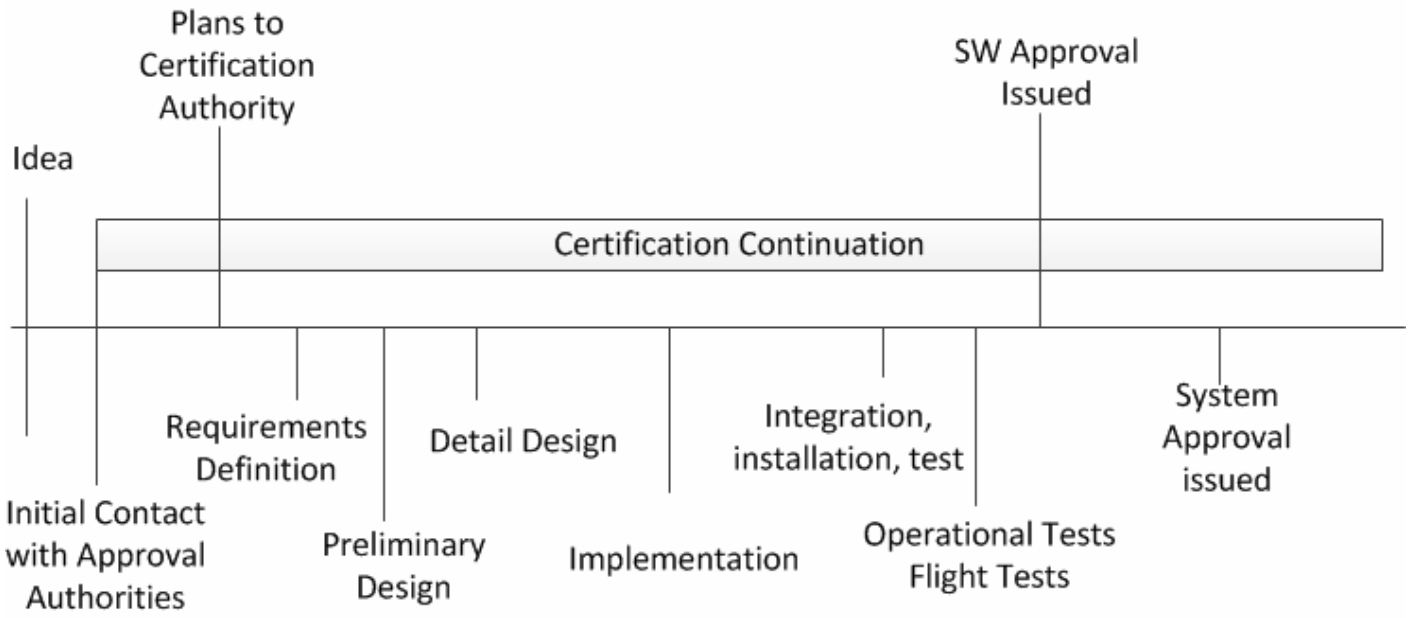
Objectives Distribution in DO-178B



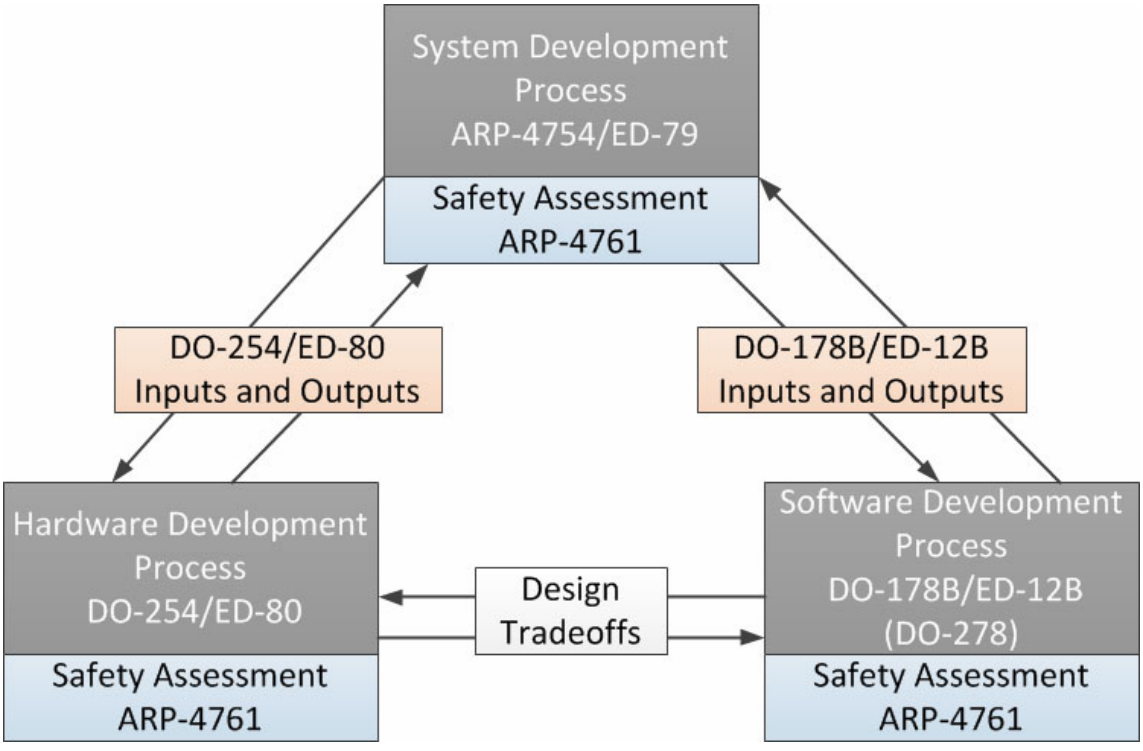
Agenda

- Introduction to DO-178B
- System Aspects
- Software Lifecycle Management
- Certification Artifacts and Techniques
- Future: DO-178C

Typical Development road plan



System Development Process



System Aspects and System Safety

- System requirements „have to be trusted” → start all over if changed
- Failure Condition Categories (Catastrophic, major, etc.)
- System Safety Assessment based on SAE ARP 4761
  - Fault Tree Analysis, Dependence Diagram, Markov Analysis, Failure mode and Effect analysis, Common Cause and mode Analysis, etc.
- SW requirements derived from System requirements → however, certain SW requirements can have impact on System requirements!

SW Safety

- SW Safety level based on potential failure conditions
  - Level A → „failure in the SW would result in catastrophic failure condition the aircraft”
- DO-178B defines the interface with the systems
- DO-178B software classes
  - User-modifiable software
    - Entertainment software
  - Option-selectable software
    - Cartography software
  - Commercial Off-The-Shelf software
    - RTOS
  - Field-Loadable software
    - Maintenance software

Agenda

- Introduction to DO-178B
- System Aspects
- Software Lifecycle Management
  - Planning
  - Development
- Certification Artifacts and Techniques
- Future: DO-178C

Software Life Cycle

- Planning should proceed all development activity
- Four building blocks :
  - Define Requirements (R)
  - Design the program (D)
  - Code the program (C)
  - Integrate the program (I)
- Allows various development sequences

Example processes:

R-D-C-I → Waterfall

R-C-I-C-I-C-I-R-D-C-I → Rapid prototyping

R-I → Previous designed SW

The plans

- Five different plans
  - SW Development Plan
  - SW Verification Plan
  - SW Quality Assurance Plan
  - SW Configuration Plan
  - SW Aspects of Certification
- Verification, management, quality assurance and certification are overlaid on the defined development process

Software Planning

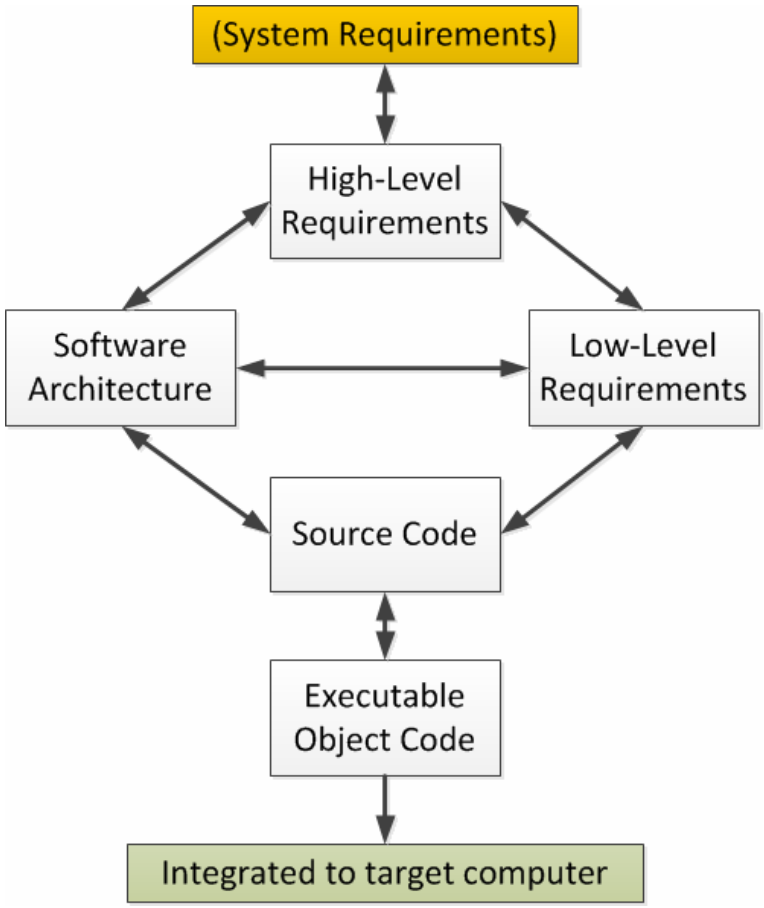
- Transition criteria
  - „the minimum conditions, as defined by the software planning process, to be satisfied to enter a process”
  - Tells when you are done and can proceed
  - Good characteristics: quantifiable, documented ☺
- Additional considerations
  - COTS
  - Previously developed components
- Environments
  - Methods and notations
  - Language with any constraints
  - Development and verification tools

Software Planning

- SW development standards
  - SW requirements standard
    - Language to be used (e.g., plain 500 English)
  - SW design standards
    - Complexity limits, exclusion of recursion, dynamic memory allocation
  - SW Code standards
    - Syntax, semantics and constraints

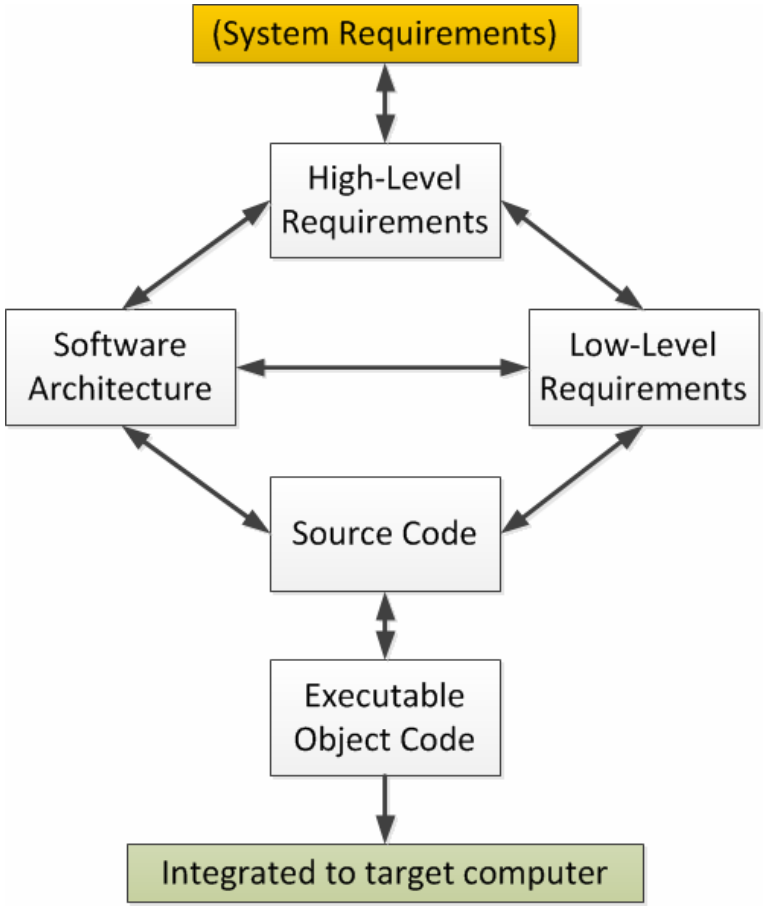
SW Development

- High-Level requirements
  - Based on system analysis and safety assessment
  - Black-box view of the software component
  - System level considerations
  - Functional requirements by mode of operation
  - Performance criteria
  - Timing requirements
  - Memory size constraints
  - HW and SW interfaces



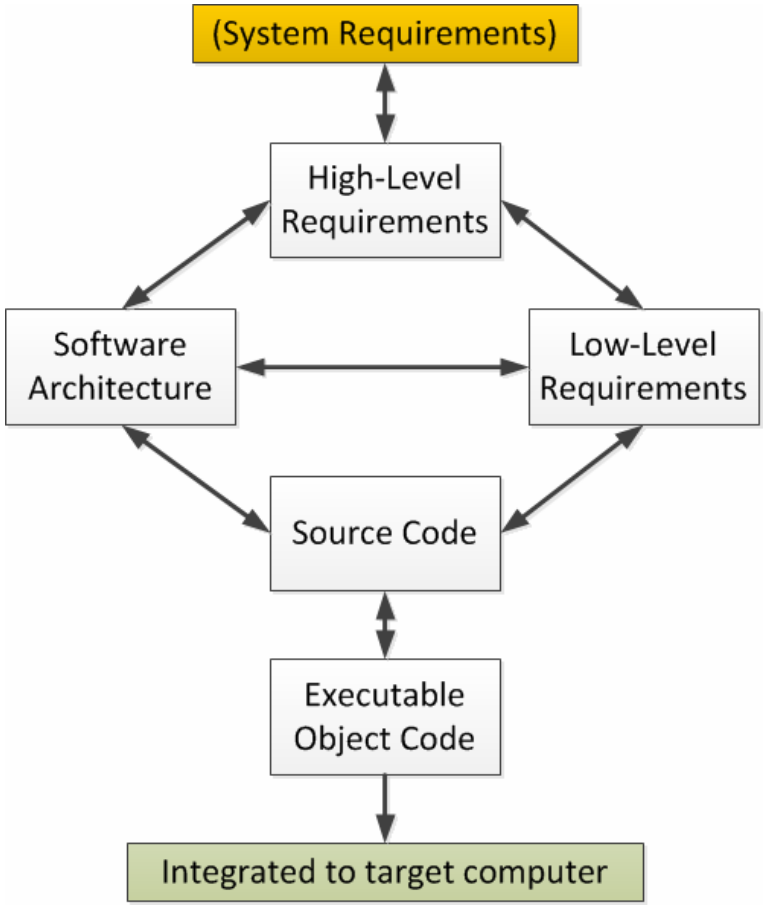
SW Development

- Low-Level requirements and Software Architecture
  - SW requirements
  - Derived from High-Level requirements
  - Design constraints
    - Task allocation
    - Algorithms
    - Data Structures
  - Input/output definitions
  - Data and Control flows
  - Resource management and scheduling (e.g., partition scheduling in ARINC 653)
  - Design Methods



SW Development

- Source Code
  - Usually collection of „high-level" language and assembly
  - Includes linker files, compile commands etc.
- Executable
  - Completely target computer specific
  - „machine readable"
- Final output is the integrated system on the target platform



Agenda

- Introduction to DO-178B
- System Aspects
- Software Lifecycle Management
- Certification Artifacts and Techniques
  - Verification
  - Configuration Management
  - Quality Assurance
  - Certification/Approval Liaison
- Future: DO-178C

Integral Process - Verification

- Two purposes
  - Demonstrate intended function
  - Demonstrate (to the extent possible) the absence of unintended function
- Consists of
  - Reviews
  - Analysis
  - Testing
- Important: The FAA or EASA representative needs to accept all part of the verification process. (e.g., test cases)

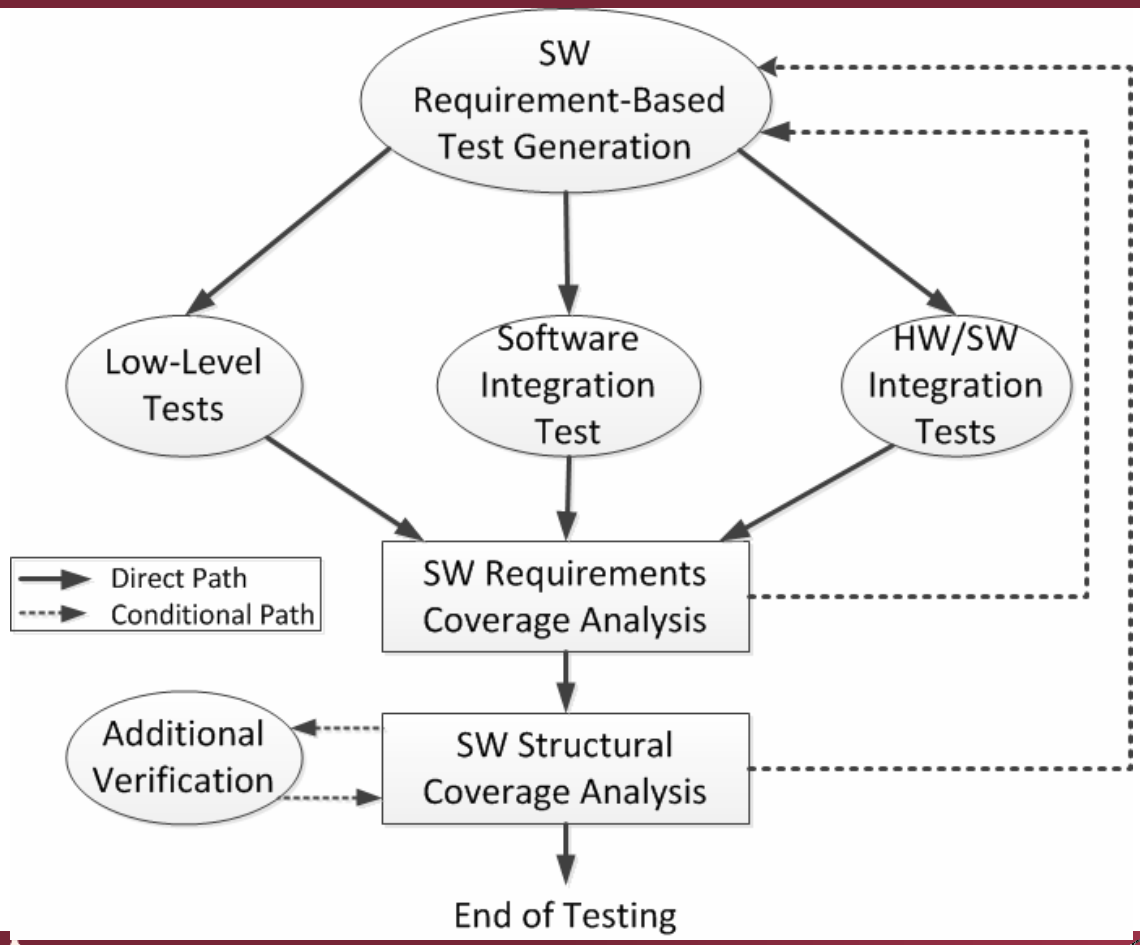
Integral Process - Verification

- Reviews
  - Qualitative assessment of the process or product
  - Typical implementation: checklist
  - Applied on all SW Development process step (HLR, LLR, SA, SC, Test cases, etc.)
- Analysis
  - Provide repeatable evidence of correctness
  - Typical implementation: timing, stack analysis, data flow and call-tree

Traceability

- Show which part of the source code fulfils which requirement
- Through the complete development cycle from requirements to source code
- Typical implementation:
  - Txt files ☹
  - Excel sheets ☹
  - Doors
- Code generator tools give good support
- Essential for maintainability

Integral Process – Verification Software Testing



Integral Process – Verification Software Testing

- Categories of Tests
  - Normal range
  - Robustness (abnormal range)
- Typical approaches
  - Equivalence Classes and Boundary Values
  - Multiple Iteration testing for time related functions
  - Testing State Transitions
  - Initialization with abnormal conditions
  - Failure modes of input data
  - Boundary values in loops, protection mechanisms

Integral Process – Verification Software Testing

- Structural Coverage
  - Determine what software structure were not exercised
- Levels:
  - Decision Coverage
  - Statement Coverage
  - Modified Decision Condition Coverage (MCDC)
    - Each decision tries every possible outcome
    - Each condition in a decision takes on every possible outcome
    - Each entry and exit point is invoked
    - Each condition in a decision is shown to independently affect the outcome of the decision
- Gaps
  - Complier induced code (e.g., array bound checks)
  - Deactivated code
  - Dead code
- Performed on source code,
  - except Level A
    - Correspondence must be shown
    - Complier optimization can introduce new code
- In addition, coverage of data and control coupling is required

Integral Process – Verification Software Testing

- IF(C AND(A OR B)) THEN Foo(); ENDIF;
- Modified Decision Condition Coverage (MCDC)
  - Each decision tries every possible outcome
  - Each condition in a decision takes on every possible outcome
  - Each entry and exit point is invoked
  - Each condition in a decision is shown to independently affect the outcome of the decision
- Use TOOLS for automated MCDC calculations

#	A	B	C	Foo Executed
1	0	0	0	NO
2	0	0	1	NO
3	0	1	0	NO
4	0	1	1	YES
5	1	0	0	NO
6	1	0	1	YES
7	1	1	0	NO
8	1	1	1	YES

Coverage Type	Minimum # of Test Cases	Possible Combinations
Statement	1	4 or 6 or 8
Decision	2	4 or 6 or 8 + Any NO
MCDC	4	2,3,4, and 6 OR 2,4,5 and 6

Integral Process – Verification Software Testing

- Coupling
  - Control coupling: the manner which one SW components influences the execution of other
  - Data coupling: the dependence of a SW component on data not exclusively under its control
- Coupling and Cohesion
  - Coupling ↑ Cohesion↓ and vice-versa
  - Modular SW components have strong cohesion (not exposed to the calling modules) → single purpose → highly reusable as keeps changes local → ”cheap” verification and certification

Integral Process – Configuration Management

- Ensures that changes are accomplished in a controlled manner
- Includes all activities for establishing configuration identification, change control and archival of data
- Multiple level
  - Highest → almost impossible to change (why do we have a planning and design phase?)
  - Lowest → easy but documented!
- Generates Problem Reports
- Control Category
  - Defines the level how information is stored and handled (remember the example table showing the „Executable Object Code compiles with low-level requirements” objective)
  - Two levels: CC1 and CC2
  - E.g., Traceability, baselines, change review, release management, change control, protection against unauthorized changes, etc.

Integral Process – Quality Assurance

- Assesses the SW life cycle process and their outputs to obtain assurance that the objectives are satisfied
- Independent checks and staff
- Works closely with development team

Integral Process – Certification/Approval Liaison

- Communication between application developer and certification authority
- Proposes compliance and obtain agreement on the plan
- Software Accomplishment Summary
  - Covers all areas
  - Legal issues also (if something goes wrong the developer is responsible!)

Additional Considerations in DO-178B

- Tool Qualification
  - Software Development Tools
    - Can introduce errors
    - Same objectives as the development process → verified on the same level as the developed application!
    - E.g., Scade Suite, Matlab Stateflow
  - Software Verification Tools
    - Can only fail to detect errors
    - Tool operation req. Must be satisfied under normal operating conditions
    - e.g., static source code analyzer ASTRÉE, CAVEAT

Agenda

- Introduction to DO-178B
- System Aspects
- Software Lifecycle Management
- Certification Artifacts and Techniques
- Future: DO-178C

DO-178C

- **DO-178C - Software Considerations in Airborne Systems and Equipment Certification**
- Awaited in 2011
- New certification for avionics software development
- Incorporates ”novel” development and verification techniques
- Core is almost the same as DO-178B but
- Dedicated subgroups
  - SG3: Tool Qualification
  - SG4: Model Based Design and Verification
  - SG5: Object-Oriented Technology
  - SG6: Formal Methods

DO-178C

- Object Oriented Technology
  - C++ and Ada
  - Safety Critical Java
  - Restricted use (deterministic behavior)
- Tool Qualification
  - Special rules for tools
  - More than two categories
- Model Based Design and Verification
  - Use of models for source code synthesis and verification
  - Early model based validation
  - Matlab Simulink (already used), AADL
  - Largest and most cumbersome subgroup ☺

DO-178C

- Formal methods
  - Already used in many projects
  - Mature technologies available
  - Defines how certification credits can be earned by its use
  - Can be part of the Development process
  - Typical tools
    - Model checker
    - Static code analyzers
    - Theorem provers (only in limited scenarios)