Disclaimer

Note that these slides are intended to give an introduction to the development of software for avionic systems and do not represent official aerospace regulatory documentation.

Recommended Reading

System Software Safety FAA Handbook Chapter 10 (read it) [FAA]
DO-178B Questions and Answers (read this one next) [High Rely Inc. 2005]
Development costs (read this one) [High Rely Inc. 2009]
Lifecycle Diagram (worth studying) [Embedded]
Software Considerations... (DO-178B) Library: shelfmark 629.1345 RAD (The definitive text, heavy going)
What defines Avionics Software?

- Adds value but not weight.
- Is (possibly) safety critical.
- Development process must be documented (legal requirement).
- Development model covers more than standard software development.

What is a System Development Lifecycle?

- From concept to disposal;
- Guidelines and standards for the whole process;
- System is certified, not its parts;
- Avionics systems are just one part;
- Avionics software is a subset of that part.
- Clearly defined deliverables for each stage.
The whole development process begins with Standards and Guidelines:

- Certification Specifications for Airworthiness of Large Aeroplanes (CS-25)
  - Performance and handling qualities (e.g., one engine inoperative)
  - Structure (e.g., gusts envelope, manoeuvres envelope, fatigue requirements, etc.)
  - Design and Construction (e.g., emergency evacuation provisions, fire protection, etc.)
  - Powerplant installation (e.g., uncontained powerplant failure, etc.)
  - Systems and equipment (e.g., systems safety analyses, requirements for electrical, hydraulic and pneumatic systems, required equipment for flight and navigation, etc.)
  - Manuals and limitations (e.g., speed limitations, flight manual, continued airworthiness manual, etc.).

Software for Civil Aviation is developed using guidelines such as:
- DO-178B / EUROCAE ED-12B: Civil Aviation USA / UK
- ARP4761, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment
- IEC 61499 standard for modelling Distributed Control Systems

For Military Aviation:
- US Defence
- DEF STAN 00-55/56

DO-178B : EUROCAE ED-12B

DO-178B ... The Detail

**Assurance**
- Defines 5 levels of Design Assurance (Effect of software fault)
  - Catastrophic, Hazardous, Major, Minor, None

**Processes**
- Defines 5 processes
  - Planning, Development, Verification, Configuration Management, QA

**Objectives**
- Defines 66 objectives
  - Planning 7, Dev. 7, Verification 40, Config 6, QA 3, Cert. 3

**Design Assurance**

What if the software goes wrong?

What will the impact be on the system(s)?

Software Level Definitions are based upon failure of the software to potentially cause or contribute to
- Level A: Catastrophic failure
- Level B: Hazardous/severe-major failure
- Level C: Major failure
- Level D: Minor failure
- Level E: No effect on operational capability or pilot workload
**Design Assurance 2**

<table>
<thead>
<tr>
<th>System safety assessment process</th>
</tr>
</thead>
<tbody>
<tr>
<td>establishes failure condition categorization (AMJ 25-1309)</td>
</tr>
<tr>
<td>identifies required assurance level (DO-178B)</td>
</tr>
<tr>
<td>Design assurance level determines level of effort required for compliance</td>
</tr>
</tbody>
</table>

**Processes**

<table>
<thead>
<tr>
<th>Review: Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
</tr>
<tr>
<td>Development</td>
</tr>
<tr>
<td>Requirements</td>
</tr>
<tr>
<td>Design</td>
</tr>
<tr>
<td>Coding</td>
</tr>
<tr>
<td>Integration</td>
</tr>
<tr>
<td>Verification</td>
</tr>
<tr>
<td>Configuration</td>
</tr>
<tr>
<td>Quality Assurance</td>
</tr>
<tr>
<td>Certificate Liaison</td>
</tr>
</tbody>
</table>

**Processes**

**Planning**

Define and coordinate the whole lifecycle

- Activities include
  - choice of development and coding standards
  - choice of methods and tools
  - coordinate revisions
  - ...
- Objectives include
  - determining sequencing of all processes
  - ensure tools/methods consistent with safety level
  - ...

**Development**

four subprocesses

- **Requirements**: Develop high-level requirements
- Objectives include:
  - high-level requirements developed
  - derived requirements fed to system safety process
  - ...
## Development

- **Design**: Develop low-level requirements and software architecture
  - Objectives and activities include:
    - low-level requirements developed
    - derived low-level requirements fed to system safety process
    - monitor control flow and data flow

- **Coding**: Implement the source code
  - Objectives include:
    - code is traceable, verifiable, consistent and correct
    - implements the low-level requirements
    - incorrect inputs detected and fed back to requirements process

- **Integration**: Hardware/software integration
  - Objectives include:
    - executable code generated from the source
    - executable code loaded onto target hardware
    - feed errors back to appropriate process

## Verification 0

- **Review...Analyse...Test** of:
  - High-level requirements
  - Low-level requirements
  - Software Architecture (design structure)
  - Source Code
  - Integration Outputs
  - Test Cases, Procedures, Results
<table>
<thead>
<tr>
<th><strong>Verification 1</strong></th>
<th><strong>Verification 2</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Review...Analyze...</td>
<td></td>
</tr>
<tr>
<td>high-level requirements</td>
<td></td>
</tr>
<tr>
<td>compliance with system requirements</td>
<td></td>
</tr>
<tr>
<td>conformance to standards</td>
<td></td>
</tr>
<tr>
<td>compatibility with target computer eg performance</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td><strong>Verification 2</strong></td>
<td><strong>Verification 3</strong></td>
</tr>
<tr>
<td>Review...Analyze...</td>
<td></td>
</tr>
<tr>
<td>low-level requirements</td>
<td></td>
</tr>
<tr>
<td>compliance with high-level requirements</td>
<td></td>
</tr>
<tr>
<td>conformance to standards</td>
<td></td>
</tr>
<tr>
<td>accuracy and consistency of low-level requirements</td>
<td></td>
</tr>
<tr>
<td>accuracy and behaviour of proposed algorithms</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td><strong>Verification 3</strong></td>
<td><strong>Verification 4</strong></td>
</tr>
<tr>
<td>Review...Analyze...</td>
<td></td>
</tr>
<tr>
<td>Source Code</td>
<td></td>
</tr>
<tr>
<td>compliance with low-level requirements</td>
<td></td>
</tr>
<tr>
<td>data-flow / control flow comply with s/w architecture</td>
<td></td>
</tr>
<tr>
<td>no undocumented functions</td>
<td></td>
</tr>
<tr>
<td>conformance to software coding standards</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td><strong>Verification 4</strong></td>
<td></td>
</tr>
<tr>
<td>Review...Analyze...</td>
<td></td>
</tr>
<tr>
<td>integration</td>
<td></td>
</tr>
<tr>
<td>Incorrect hardware addresses</td>
<td></td>
</tr>
<tr>
<td>Memory overlaps</td>
<td></td>
</tr>
<tr>
<td>Missing Software components</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
...Test
Compliance
demonstrate that software satisfies its requirements
Confidence that
no unacceptable failure conditions due to errors
robust wrt abnormal inputs and conditions

Interlude.
Distribution of Software Errors. DeMarco 1978

Requirements 56%
Design 27%
Other 10%
Code 7%

Cost of Rectifying Software Errors (DeMarco '75)

Interlude

Interlude...
**Configuration Management**

Runs concurrently with entire service life of product

- Activities include
  - Identification of all controlled components
  - Change control
  - Archiving
  - ...

- Objectives include
  - ensure changes are reviewed and recorded
  - ensure physical archiving and recovery
  - ensure problems receive attention
  - ensure changes are implemented
  - ...

**Quality Assurance**

QA runs throughout the development lifecycle

- Activities include
  - tracking deviations from plans
  - provide assurance of planning consistency
  - produce records of QA activities
  -...

- Objectives include
  - ensure transition criteria between stages satisfied
  - ensure lifecycle processes are complete
  - processes comply with plans and standards

**Certification Liaison**

Ongoing throughout the software life cycle

- establish communication between Developer and Authority
- agree plan for software aspects of certification
- provide evidence: processes satisfy the plans

**Objectives by Level**

- Satisfied with independence
- Satisfied

- Software Assurance Level

![Chart showing satisfaction levels across software assurance levels](chart.jpg)
**Avionic Software development is a lengthy and complex process**

- Effort depends on assurance level
- Should follow guidelines
- Systems comprise 30% of costs
- Embedded Systems nearly half of that
- More than €100 per line for level A

**References**