Quality in Aviation Software

Chris Hartgroves C.Eng. CQP
Design Assurance
SELEX Galileo
Contents

- Introduction
- Terminology
- Historical context
- Poor-quality aerospace software
- Governance
- Standards
- DO-178 ‘Software aspects of Certification’
- Engineering lifecycles
- Software engineering
Overview

- What is ‘Quality in Aviation Software’, why have it?
- Governance and Standards
- Software Engineering
# Who am I?

<table>
<thead>
<tr>
<th>Companies / industries</th>
<th>Qualifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1975-1985: various small companies – sonar, industrial controls, petro-chem, satcoms</td>
<td>Computer Science degree</td>
</tr>
<tr>
<td>Analogue &amp; digital circuit design</td>
<td>(Manchester University)</td>
</tr>
<tr>
<td>1985: Product Manager for <em>early</em> PCs (Olivetti)</td>
<td>C.Eng MIEE (now MIET)</td>
</tr>
<tr>
<td>1988: Racal / Thales Simulation &amp; Avionics SW Engineer, SW Project Leader, Product</td>
<td></td>
</tr>
<tr>
<td>Group Manager.</td>
<td></td>
</tr>
<tr>
<td>2007: Logica Space &amp; Defence group SW assurance and safety manager</td>
<td></td>
</tr>
<tr>
<td>2009: Selex Galileo – defence electronics - airborne, battlefield, &amp; homeland security;</td>
<td></td>
</tr>
<tr>
<td>SW Quality Assurance</td>
<td></td>
</tr>
</tbody>
</table>
Typhoon Cockpit
Terminology –

- What is Aviation Software?

- What constitutes Software Quality?

- What is Software Assurance?
Terminology - What is Aviation Software?

Software in Aeroplanes

- Flight or Mission Critical – e.g. fly-by-wire, engine control, radar, defensive aids
- Health & Safety related – e.g. environment control (pressure, temperature, humidity, lighting, sustenance)
- Workload-reducing – e.g. secondary systems using sophisticated displays
- Inconsequential for aircraft safety, e.g. In-Flight Entertainment, passenger communications

Ground-Based software affecting safety of flight

- Air Traffic Management
- GPS-based navigation
- Position reporting

Ground-Based with little or no safety implications

- Airline passenger and goods / freight management systems
- Airline logistics
Could an in-flight entertainment system be safety-critical?

Yes - if it causes an electrical fire.

Following this accident the FAA issued an order that prohibited installation of a certain In-flight Entertainment Network system.
What is Software Quality?

Quality of Product, and (therefore) of the Engineering and Assurance processes and personnel:

- Fitness for purpose - meeting the correct set of requirements to deliver the desired outcome
- Delighting customers; delivering more than they expected (requirements, costs, schedule)
- Meeting other stakeholders’ aspirations
- Setting standards, being best in class
- Participating in industry, national, and international groups to advance the state of the art
What is Software Quality Assurance?

- ‘systematic monitoring and evaluation of the various aspects of a project, service or facility to maximize the probability that minimum standards of quality are being attained’ *Websters dictionary*

- “the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle, and that the software functions in the intended manner.” *US National Information Assurance Glossary*
1\textsuperscript{st} gratuitous (fabulous) aeroplane photo
Historical Context

The demand for functional complexity, reliability and low-mass has driven the extensive use of computers.

Systems that could affect safety must be certified, to defined standards; national standards for domestic flights, and to international standards.

Aviation Software requires –

- A systematic and clear (verifiable & auditable) way of demonstrating the safety of software
- Means of assessing the consequences of SW defects
- International standardisation to allow international air-travel
Historical context

A380 Development cost = €12bn

• Software cost = €4bn
Failures and Defects in Software

Software cannot ‘fail’ – there are no failure mechanisms; however, SW can be ‘wrong’ in 3 ways:

- Accidentally defective
- Sabotaged – deliberately introduced wrong or undesirable functionality
- Correctly implementing the wrong requirements (where ‘wrong’ may be ‘incomplete’)

Eclipse 500 Cockpit
The switch & gauge intensive zero-software Concorde cockpit
Consequences of poor-quality aerospace software

No ‘hull losses’ attributable to SW faults to date

Examples of -

- Commercial ‘disasters’
- Space example
- ‘Implicated’ SW
Ariane Flight 501 – the billion-dollar bug

Ariane Flight 501 - (4/6/96) The failure was caused by the complete loss of guidance and attitude information due to specification and design errors in the software of the inertial reference system. Software was re-used without adequate verification.
Rounding Error – Patriot

- February 25th 1991, during the Gulf War
- A Scud struck an American Army barracks and killed 28 soldiers.
- Cause was an inaccurate calculation of the time due to computer arithmetic errors (truncation of 1/10 second).
- Patriot battery had been up around 100 hours
- Resulting time error due to the magnified chopping error was about 0.34 seconds.
- A Scud travels at about 1,676 meters per second ...
- The Defensive System should have been reset regularly to prevent ‘clock-creep’, but...
Who’s who – civil and military?
- Standards hierarchy
- RTCA / Eurocae DO-178B / ED-12B
Civil air-safety players

- United Nations
  - IMO = International Maritime Organisation
  - ICAO = International Civil Aviation Organisation – SARPS

- Civil Aviation National & International Authorities –
  - CAA, FAA,
  - Technical Standards bodies -
    - RTCA, members include AEEC, DoD, Boeing, Lockheed-Martin etc.
    - EUROCAE (‘rubber-stamps’ RTCA products, e.g. ED-12B is DO-178), members include CAA, Inmarsat, Qinetiq, Thales, Ultra, etc.
    - European Aviation Safety Agency (EASA, EU agency)
Governance: Military air-safety players, Certifiers / approvals

Military Aviation Authority (MAA)

Enhancing the delivery of operational capability through continuous improvement in military Air Safety, associated culture, regulation and practice

MAA Regulatory Publications - Regulatory Articles and Supporting Manuals - are now available.
Please see link to MAA Documents under 'Related Pages'.

Part of the Ministry of Defence (MOD), the MAA is an independent and autonomous organisation responsible for the regulation, surveillance, inspection and assurance of the Defence Air operating and technical domains. It ensures the safe design and use of military air systems.

As the single regulatory authority responsible for regulating all aspects of Air Safety across Defence, the MAA has full oversight of all Defence aviation activity. Through independent audit and continuous surveillance of military aviation, the MAA aims to provide the Secretary of State for Defence (SofS), through the 2nd Permanent Under Secretary of State for Defence (2nd PUS), the necessary assurance that appropriate standards of Air Safety are maintained in delivering operational capability.

The MAA draws the authority to discharge its regulatory role from a Charter signed by SofS. The Charter also specifies the MAA's high level governance arrangements and broad responsibilities.
Standards & Process Models / Philosophy

ISO 9001 – ‘universal’ Quality Management

TickIT (& plus) ‘A quality-management certification program for software development,’

CMMI (CMM) – ‘a process improvement approach’

SAE: AS, ARPs & RTCA DOs for Civil Aviation

ISO – International Organisation for Standardisation

CMMI - Capability Maturity Model Integration

SAE – once was USA ‘Society for Automotive Engineers’, now much wider remit, encompassing– Aerospace Standards (AS), Aerospace Recommended Practices (ARP).

RTCA – once was USA ‘Radio Technical Commission for Aeronautics’: produces DOs (origin lost in time) for Civil Aviation

Presently there will be a third stage, BS EN 9115, providing a further level of tailoring for Deliverable Software -
Characteristics of the Maturity levels

- **Level 1: Initial**
  - Processes unpredictable, poorly controlled and reactive

- **Level 2: Managed**
  - Process characterized for projects and is often reactive.

- **Level 3: Defined**
  - Process characterized for the organization and is proactive.
    (Projects tailor their process from organization’s standard)

- **Level 4: Quantitatively Managed**
  - Process measured and controlled

- **Level 5: Optimizing**
  - Focus on process improvement

Capability Maturity Model Integration (CMMI)
Civil Aviation Certification Standards Hierarchy

- SAE ARP 4761: Safety Assessment Process Guidelines & Methods
- SAE ARP 4754: System Development Process
- DO-254: Hardware
- DO-178B: Software

Increasing level of detail as ‘system’ decomposed from whole aircraft.
DO-178B / ED-12B
“Software Consideration in Airborne Systems and Equipment Certification”

Primary SW certification standard for civil aviation
DO-178B is the primary tool used in the certification of flight software. It has achieved its pre-eminence through being –

- Non-prescriptive,
- Slow-cooked by an inclusive ‘public’ committee,
- Established – 178B since 1992,
- ‘The only game in town’ – FAA 8110.49 only recognises DO-178B route to software approval.

+ the military move to adoption of COTS equipment and standards.

DO-178C has been similarly ‘slow-cooked’ and may be issued in 2011 ...
DO-178B was created by a committee with representatives from –

- Aircraft builders, operators & owners;
- Avionics and aircraft systems designers / manufacturers;
- Certifiers, administrators.

DO-178B embraces the philosophy that –

- The earlier faults are discovered and rectified, the cheaper the fix;
- Software quality can not be assured solely by testing the product; it must be built-in to every stage of the development, from project planning and requirements capture;
- Individuals and Groups can become ‘blind’ to their own systematic faults; independent assurance is needed.
## Classifying the safety significance of SW

<table>
<thead>
<tr>
<th>DO-178B Software Level</th>
<th>What happens if software of this level ‘behaves anomalously’?</th>
<th>Probability (to be proven)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Catastrophic</td>
<td>&lt;10E-9</td>
</tr>
<tr>
<td></td>
<td>Plane crashes, everyone dies.</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Hazardous / Severe-Major</td>
<td>&lt;10E-7</td>
</tr>
<tr>
<td></td>
<td>Some may die, but not the flight crew.</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Major</td>
<td>&lt;10E-5</td>
</tr>
<tr>
<td></td>
<td>Damaged aircraft, injuries.</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Minor</td>
<td>&gt;10E-5</td>
</tr>
<tr>
<td></td>
<td>Manageable – increased workload for crew.</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>No effect</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>No effect</td>
<td></td>
</tr>
</tbody>
</table>
DO-178B -
Software Levels (Design Assurance Levels)
Ten tailoring tables -
<table>
<thead>
<tr>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  Software Planning (7 items)</td>
</tr>
<tr>
<td>2  Software Development Process (7 items)</td>
</tr>
<tr>
<td>3  Verification of Outputs of Software Requirements Process (7 items)</td>
</tr>
<tr>
<td>4  Verification of Outputs of Software Design Process (13 items)</td>
</tr>
<tr>
<td>5  Verification of Outputs of Software Coding &amp; Integration Process (7 items)</td>
</tr>
<tr>
<td>6  Testing of Outputs of Integration Process (5 items)</td>
</tr>
<tr>
<td>7  Verification of Verification Process Results (8 items)</td>
</tr>
<tr>
<td>8  Software Configuration Management Process (6 items)</td>
</tr>
<tr>
<td>9  Software Quality Assurance Process (3 items)</td>
</tr>
<tr>
<td>10 Certification Liaison Process (3 items)</td>
</tr>
</tbody>
</table>
DO-178B level tailoring

<table>
<thead>
<tr>
<th>Subject</th>
<th>Notes</th>
</tr>
</thead>
</table>
| **4** Verification of Outputs of Software Design Process (13 items) | Level D – 1 requirement  
Level C – 9 requirements  
Level B – 13 requirements, 3 with independence  
Level A - 13 requirements, 6 with independence |
| **5** Verification of Outputs of Software Coding & Integration Process (7 items) | Level D – N/A  
Level C – 6 requirements  
Level B – 7 requirements, 1 with independence  
Level A - 7 requirements, 3 with independence |
## Increasing rigour with SW Level – test coverage

<table>
<thead>
<tr>
<th>Level</th>
<th>Test coverage</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>High-level requirements coverage</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>+ procedures and results verified, Low-level requirements coverage, Statement coverage, Software structure coverage (data and control coupling).</td>
<td>Statement coverage = every line of source code must have been visited. Host-based to allow exception provocation, etc.</td>
</tr>
<tr>
<td>B</td>
<td>+ decision coverage (all conditional outcomes)</td>
<td>Explore every option of IF / ELSE, CASE switches etc (including defaults)</td>
</tr>
<tr>
<td></td>
<td>Decision, statement and structure coverage must be ‘satisfied with independence’.</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>+ ‘MCDC’ – Modified Condition / Decision Coverage</td>
<td>Show the effect of every term in every Boolean condition expression.</td>
</tr>
<tr>
<td></td>
<td>All 8 items require independence.</td>
<td></td>
</tr>
</tbody>
</table>
Software Engineering demystified
(All you ever wanted to know, but were afraid to ask?) -

- Lifecycle modelling
- Development activities
- Examples of Tools & Methods
1) A (standard) SW lifecycle model
2) Putting the SW lifecycle model into context within a system development lifecycle
3) Identifying development phases, and phase-independent activities
4) Some examples of detailed method for -
   a) Requirements definition
   b) Requirements analysis
   c) High- and Low-Level Design
   d) Source Code
SW Engineering Process - simplified
Software Engineering – V-Model lifecycle

- Requirements
- High-Level (or ‘Architectural’) Design
- Low-Level / detailed Design
- Source Code
- Object (Compiled) Code
- Synthesis / Integration

- Qualification
- System integration
- Sub-System integration
- Module integration

Analysis / Decomposition
The Systems Engineering V-Model

This diagram gives examples in light blue text of typical major Work Products and the activities where they are generally produced.

One of these Subsystems is Software
# DO-178B ten tailoring tables

<table>
<thead>
<tr>
<th>Subject</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  Software Planning</td>
<td>(7 items)</td>
</tr>
<tr>
<td>2  Software Development Process</td>
<td>(7 items)</td>
</tr>
<tr>
<td>3  Verification of Outputs of Software Requirements Process</td>
<td>(7 items)</td>
</tr>
<tr>
<td>4  Verification of Outputs of Software Design Process</td>
<td>(13 items)</td>
</tr>
<tr>
<td>5  Verification of Outputs of Software Coding &amp; Integration Process</td>
<td>(7 items)</td>
</tr>
<tr>
<td>6  Testing of Outputs of Integration Process</td>
<td>(5 items)</td>
</tr>
<tr>
<td>7  Verification of Verification Process Results</td>
<td>(8 items)</td>
</tr>
<tr>
<td>8  Software Configuration Management Process</td>
<td>(6 items)</td>
</tr>
<tr>
<td>9  Software Quality Assurance Process</td>
<td>(3 items)</td>
</tr>
<tr>
<td>10 Certification Liaison Process</td>
<td>(3 items)</td>
</tr>
</tbody>
</table>
# Software Development Activities

<table>
<thead>
<tr>
<th>Requirements Definition (SSS), Generate SRD</th>
<th>Architectural Design</th>
<th>Detailed Design</th>
<th>Implementation, coding</th>
<th>Implementation, Integration</th>
<th>Validation and Delivery, Next level integration debugging</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Verification</th>
<th>Data Management</th>
<th>Software Configuration Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review Plans Requirements review &amp; trace matrix</td>
<td>High-level design review &amp; trace matrix</td>
<td>Review, static analysis, unit test</td>
</tr>
<tr>
<td>Integrated Test</td>
<td>Acceptance Test</td>
<td></td>
</tr>
</tbody>
</table>
SW Engineering: some methods – requirements / databases

1 Input Preparation
- RSK1: Check list
  - Scenario: Wrong list
  - Business Impact: High
  - Likelihood: High
  - Classification: 0

2 Dispensing
- RSK4: Filling Preparation
  - Scenario: Air/Particles loss of fluid
  - Business Impact: High
  - Likelihood: High
  - Classification: 1

- RSK5: Inspection
  - Scenario: Wash, wrong reply of operator
  - Business Impact: High
  - Likelihood: Low
  - Classification: 2

5 Intrusion
- D-6: Apparently unauthorised entry into the Protected Household (external event)

- D-5: Detect
  - Action by Household Alarm to identify a significant external event e.g. Intrusion, Tampering

- D-4: Timeout period
  - Period of TBD 30 seconds to allow householders to enter or exit

- D-3: Deactivate
  - Action (by user) to enable the Household Alarm
SW Engineering: some methods – requirements / tracing
Requirements Tracing

Traceability Explorer

Trace types: Traces into

[NEED-2893] Customer Management
- Traces into
  - Traces into
    - [UC-2835] Register Customer
    - Traces into
      - Traces into
        - [MOD-1058] Manage Customer Form
        - Traces into
    - [SREQ-2310] The system shall allow users to add new Customers
    - Traces into
    - [SREQ-2312] The membership card shall have dimensions of standard Credit/ATM Card
    - Traces into
    - [SREQ-2311] The system shall print a membership card
    - Traces into
  - Traces into
    - [FEAT-2884] Add Customer
    - Traces into
  - Traces into
    - [FEAT-2885] Edit Customer details
    - Traces into
  - Traces into
    - [FEAT-2886] Delete Customer
    - Traces into
  - Traces into
    - [FEAT-2887] Inactivate Customer
    - Traces into
  - Traces into
    - [FEAT-2883] Manage Customers
    - Traces into
  - Traces into
    - [FEAT-2890] Produce a list of all Customers
    - Traces into
Requirements Analysis / Use Case diagram
Requirements analysis into Design
UML ‘swim lane’ sequence diagram
Architectural design – Activity diagrams
Enrolling at College -
Architectural design – Activity diagrams
Enrolling at College – organised by Actors
Architectural design – Activity diagrams
Despatching delivery trucks

Diagram:
- Insert truck into truck queue
- [At least one van carrier is idle]
- [All van carriers are busy]
- «Remove»: Remove van carrier from idle/VC queue
- «Queue»: Idle van carriers [waiting]
- «Queue»: Trucks [waiting]
- «Remove»: Remove truck from truck queue
- «Create»: Create service and event
- «Schedule»: Schedule van carrier & service end event

End point.

Note: The diagram illustrates the process of despatching delivery trucks, including the decision points and actions taken based on the status of van carriers.
import java.awt.image.IndexColorModel;
import java.awt.image.ColorModel;
import java.awt.image.MemoryImageSource;
import java.awt.event.*;

/** The representation of a Chemical .xyz model */
class XYZChemModel {
    float vert[];
    Atom atoms[];
    int tvert[];
    int ZsortMap[];
    int nvert, maxvert;

    static Hashtable atomTable = new Hashtable();
    static Atom defaultAtom;
    static {
        atomTable.put("c", new Atom(0, 0, 0));
        atomTable.put("h", new Atom(210, 210, 210));
        atomTable.put("n", new Atom(0, 0, 255));
        atomTable.put("o", new Atom(255, 0, 0));
    }

    E:WHGBBYWSample files\Sample.Form(3523,20): * It is done to avoid enormous problems with t
E:WHGBBYWSample files\Sample.Form(5476,37): * The use of the function distrib avoids the
E:WHGBBYWSample files\Sample.Form(5591,6): * avoid a temporary anomalous number of powers
50 occurrence(s) have been found.
Verification activities (V-Model right side)

- **Requirements**
- **High-level Design**
- **Low-level Design**
- **HW / SW integration**
- **Integration**
- **Coding / Implementation**
- **Testing**

**Testing**
- **Testing**
- **Testing**

- **Demonstrate that the functional requirements have been met.**
- **Exercise the component interfaces**
- **Compiler debugging**
  - Static Analysis
  - Code Inspection (human / peer review)
  - Unit Test (does the code match the detailed design)
What is Software Quality?

- Quality of Product - Fitness for purpose
- Delighting customers, meeting other stakeholders’ aspirations
- Setting standards, being *best in class*, participating
- Best-in-class Engineering and Assurance processes and personnel
Quality in Aviation Software

Thank-you for your attendance and attention.

chris.hartgroves@selexgalileo.com