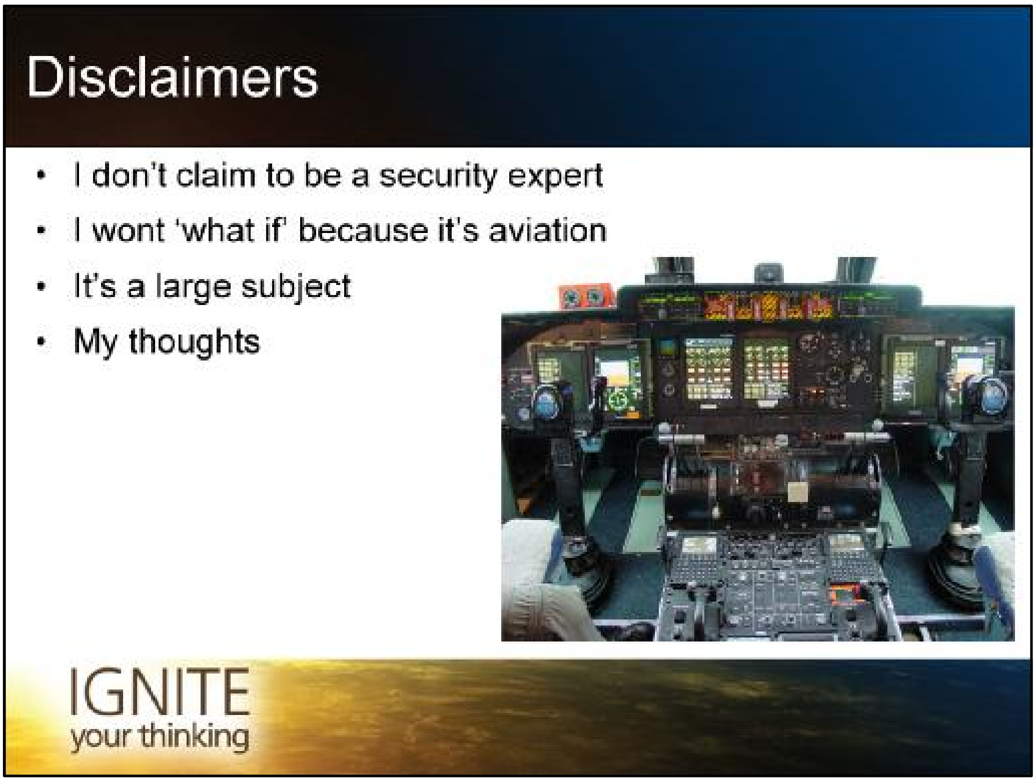
This presentation outlines the contribution of the design methods used in aircraft system development and their contribution toward security.
This includes an overview of aircraft system design requirements and design rules, how the system and software 'Design Assurance Level' is determined,
and an overview of the Software 'Assurance Level' requirements used in software development.
I'll provide my personal thoughts on the Design Assurance Level contribution to security including the inherent aircraft system design principles and processes that contribute to security,
and some thoughts on augmenting these practices with the Common Criteria requirements.



•I don't claim to be a security expert
- •Studying toward CISSP
- •I'm more about systems integrity

•Aviation!
- •Examples used are generic and do not apply to any specific system or equipment
- •I will only discuss strengths that contribute to security – not weaknesses (no "what if")
- •I will not discuss weaknesses
- •Examples used and shown are generally not available to the public. (military aircraft).

•Other things
- •I only have 30 mins to talk on a very large subject
- •This is a general talk, not scientific, general information only
- •Images used are all commons licenced
- •Personal views, not the official view of Beca Ltd.

Image: https://upload.wikimedia.org/wikipedia/commons/9/9f/C-5M_Cockpit.jpg - wikimedia commons

Aircraft system safety requirements

- The system 'design risk' categorisation
- Look at an integrated digital instrument
- Functional Hazards (risk)
- Design Assurance Level allocation

Software development 'objectives'

- What are those things that you need to do when developing software for aircraft

Contribution to security

- My view of the contributions toward security that are inherent in the overall approach

Other thoughts

https://upload.wikimedia.org/wikipedia/commons/6/63/F-CK-1_cockpit.jpg - wikimedia commons

The genesis of these delineations by weight etc, is basically linked to an individual's ability to accept risk (if you get on a small plane you can see the whole thing before boarding and have the choice not to fly if it all looks a bit dodgy, but you board a large transport through an air-bridge and seldom even see the captain).

Equally a microlight crash – with a single occupant and the aircraft not allowed to fly over built up areas has far less impact than a 747 hitting a city.

So the security comparison here is that design standards incrementally lift as more users enter the system (for rule parts),

https://c1.staticflickr.com/7/6127/5919032083_9a8238b61d_b.jpg

Photo by Mark Jones Jr.
https://www.flickr.com/photos/multiplyleadership/5919032083 – Creative Commons 2.0 (commercial and re-use)

I believe the 1309 rule is the most important design rule…

Every foreseeable operating condition – in the eyes of a sceptical regulator from the FAA whose probably seen everything. Your view of "that will never happen".. They've probably seen it happen.

§ 25.1309 Equipment, systems, and installations.
(a) The equipment, systems, and installations whose functioning is required by this subchapter, must be designed to ensure that they perform their intended functions under any foreseeable operating condition.
(b) The airplane systems and associated components, considered separately and in relation to other systems, must be designed so that—
(1) The occurrence of any failure condition which would prevent the continued safe flight and landing of the airplane is extremely improbable, and
(2) The occurrence of any other failure conditions which would reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions is improbable.

(c) Warning information must be provided to alert the crew to unsafe system operating conditions, and to enable them to take appropriate corrective action. Systems, controls, and associated monitoring and warning means must be designed to minimize crew errors which could create additional hazards.
(d) Compliance with the requirements of paragraph (b) of this section must be

shown by analysis, and where necessary, by appropriate ground, flight, or simulator tests.
The analysis must consider—
(1) Possible modes of failure, including malfunctions and damage from external sources.
(2) The probability of multiple failures and undetected failures.
(3) The resulting effects on the airplane and occupants, considering the stage of flight and operating conditions, and
(4) The crew warning cues, corrective action required, and the capability of detecting faults.
(e) In showing compliance with paragraphs (a) and (b) of this section with regard to the electrical system and equipment design and installation, critical environmental conditions must be considered.
For electrical generation, distribution, and utilization equipment required by or used in complying with this chapter, except equipment covered by Technical Standard Orders containing environmental test procedures, the ability to provide continuous, safe service under foreseeable environmental conditions may be shown by environmental tests, design analysis, or reference to previous comparable service experience on other aircraft.
(f) EWIS must be assessed in accordance with the requirements of §25.1709.

https://upload.wikimedia.org/wikipedia/commons/1/16/A330_Ditching_Button.jpg - wikimedia commons

You use the first standard to define the hazards and their severity, and the second book to develop software to meet the hazard requirements.
Really well established processes.

Don't even think about writing software for aircraft without understanding these standards.

https://upload.wikimedia.org/wikipedia/commons/b/b0/C-05_Agusta_A.109E_Carabineros_De_Chile_Flight_Deck_(8185321878).jpg Wikie

Example digital instrument

Aspect	Definition	Source
Attitude	Which way is up	Gyroscope / IRS
Altitude	How high are you?	Pitot static / Radar Alt
Heading	Where are we going?	Compass / IRS
Airspeed	How fast?	Pitot static system
VSI	up or down?	Pitot static system
Inclinometer	Are we slipping?	Spirit level

IGNITE
your thinking

Attitude = Gyro or IRS – may be built in, may be connected to a dedicated Inertial Nav, with dedicated data interface over ARINC 429 - self-clocking, self-synchronizing data bus protocol (Tx and Rx are on separate ports.
Altitude = Pitiot static system, sometimes through an air-data computer, sometimes the pipes go directly into the back of the instrument.
Heading = typically from a Flight management system – may be a 3 wire analogue syncro or ARINC 429 data word

Something to note… there are very few external interfaces. – limited or no buttons, very limited data in / out.

They might be connected to an Integrated GPS / INS, might be connected to a piece of navigation equipment,
however as we'll discuss it is robust against incorrect messages being sent.

Failure criteria

Failure condition	No more than x per flight hr	Example Functions	Design Assurance Level
Catastrophic	10^{-9}	Flight controls / primary flt displays	A
Hazardous	10^{-7}	FMS, Nav/Com	B
Major	10^{-5}	Radio Controller	C
Minor	$>10^{-5}$	Maintenance / IFE H/W	D
No Safety Effect	N/A	IFE / Galley Services	E

IGNITE
your thinking

Failure condition → Functional Hazard

Assuming backup instrument is present, imagine the pilot and co-pilot instruments like the one shown have:

- Simultaneous loss, or
- Simultaneous incorrect information

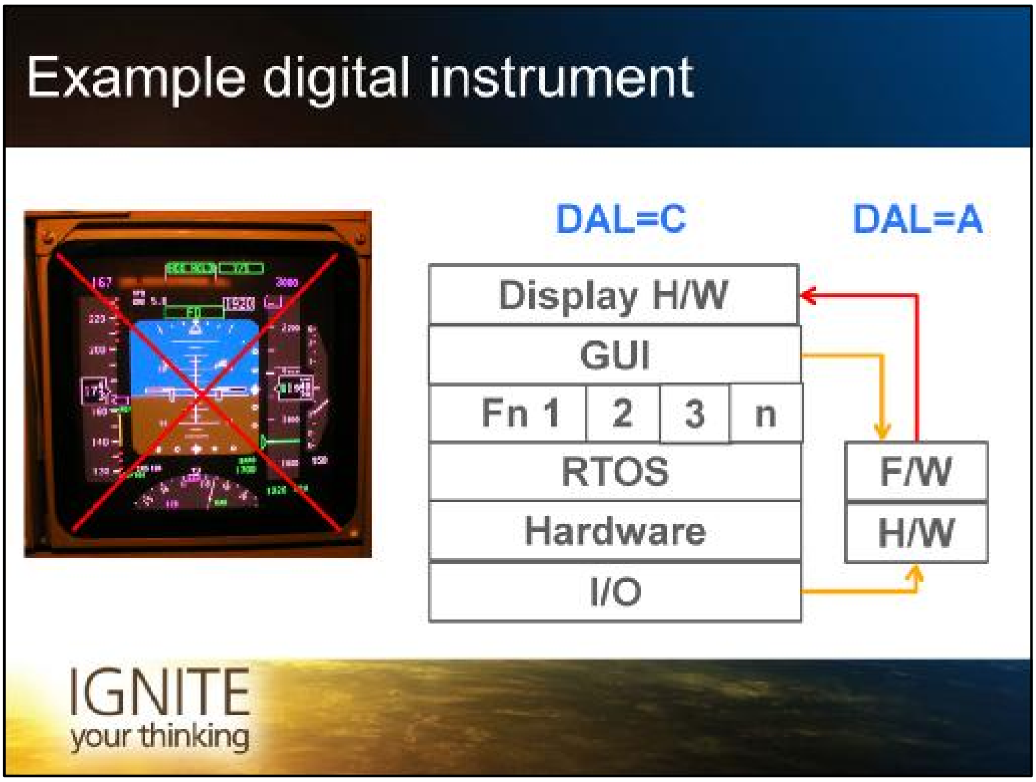
- Loss = Level C hazard
- Incorrect = Level A hazard

IGNITE
your thinking

Design Assurance Level allocation

- Level C → for loss
- Level A → for misleading

IGNITE
your thinking



I/O could be physical interfaces such as pitot static transducer, or a direct data link to a the hardware that does that.. The principles apply.

- The hardware will typically be a really well established real time processor designed for aviation…
- The RTOS will be something pretty obscure, like Greenhill's Integrity RTOS, or potentially Vxworks. These come fully documented with very tight and traceable resource allocation, which need to be validated by the instrument manufacture during testing – every line of code in the OS that is not able to be exercised has to be removed.
- There might be multiple functions, and each is separated by the OS. This includes memory allocation and timing, which must be predictable.
- I've drawn the graphical component and the display hardware at the top, to illustrate a point more than represent the actual stack.

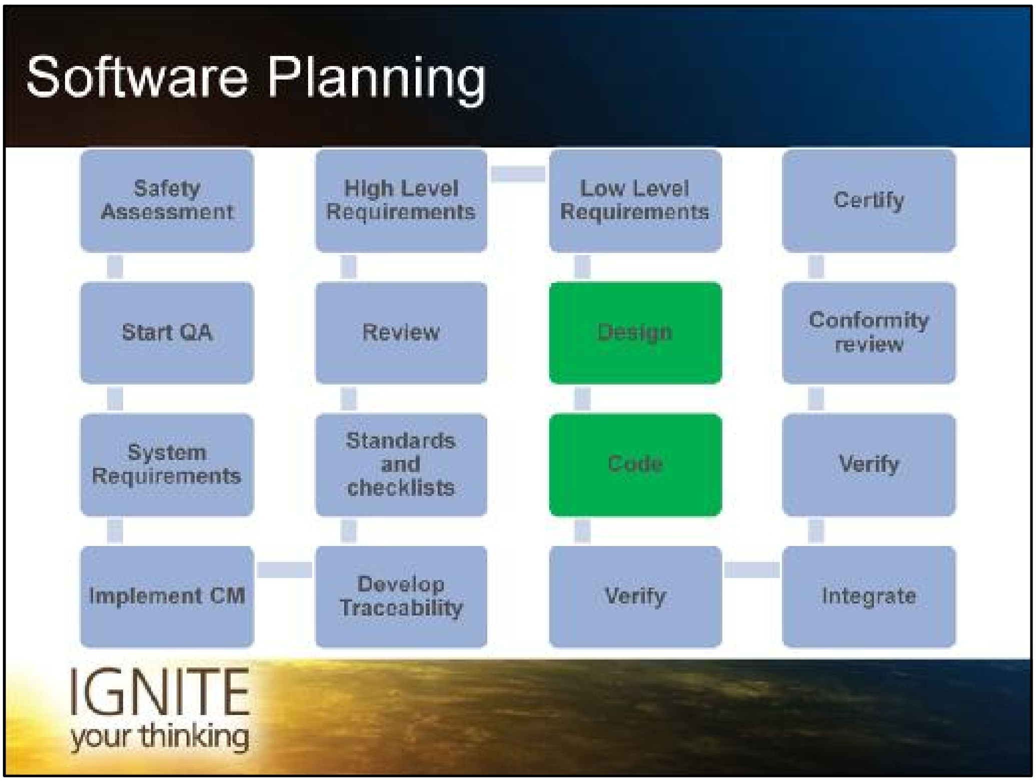
Software Development Objectives

- Objectives within RTCA DO-178C

Level	Failure condition	Objectives[1]	With independence
A	Catastrophic	71	33
B	Hazardous	69	21
C	Major	62	8
D	Minor	26	5
E	No Safety Effect	0	0

Note level E has no objectives

IGNITE
your thinking



Based on Time and development and correctness phases slide by Vance Hilderman – Highrely

Level D Objectives

- Plans (5)
- High level requirements
- Architecture developed
- Executable code developed
- Parameter data item files (if needed) verified
- Some review and analysis of high level requirements
- Review architecture if partitioning
- Normal and robustness testing of high level requirements
- Testing to verify target compatibility
- Configuration Management
- Quality Assurance (including compliance to plans)
- Accomplishment summary and configuration index

IGNITE
your thinking

Level C Objectives

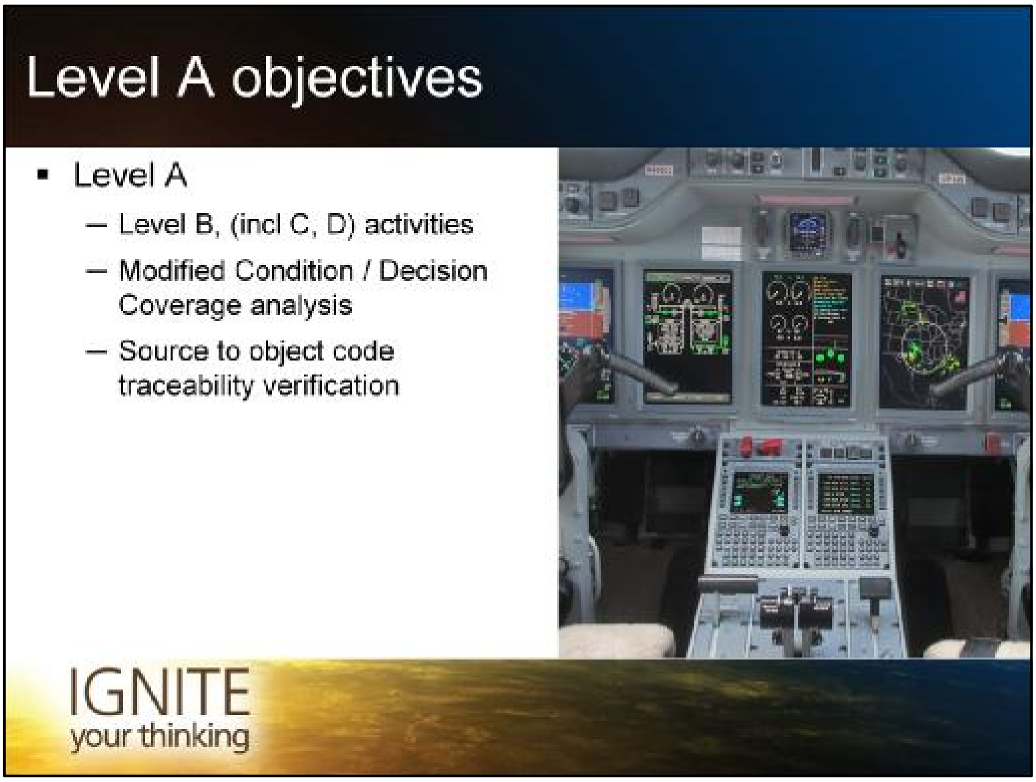
- All level D activities
- Development standards (3)
- Low level requirements developed
- Trace data developed
- Source code developed
- Additional review of high level requirements
- Some review and analysis of low level requirements
- Some review and analysis of architecture
- Review and analysis of source code
- Verification of parameter data item files
- Normal and robustness testing of low-level requirements
- Review test procedures
- Review test results
- Statement coverage analysis
- Data control and coupling analysis
- Additional QA (review plans and standards, compliance to standards, transition criteria)

IGNITE
your thinking

Level B Objectives

- Level C (incl. Level D) activities
- Decision Coverage
- Additional review and analysis of high level requirements
  - target compatibility
- Additional review and analysis of low level requirements
  - target compatibility and verifiability
- Additional review and analysis of architecture
  - target compatibility and verifiability
- Additional review and analysis of source code
  - Verifiability

IGNITE
your thinking



MC-DC Every point of entry and exit in the program has been invoked at least once, every condition in a decision in the program has taken all possible outcomes at least once, and each condition has been shown to affect that decision outcome independently.

https://upload.wikimedia.org/wikipedia/commons/a/a6/Hawker_4000_cockpit.jpg - Wikimedia commons

Basically….

- A whole lot more effort for those things that pose risk

IGNITE
your thinking

separation of impact of failure conditions – areas that pose risk are separate from those that do not, to eliminate pathways from weak areas to areas that require strong protection. It's Ok to send data from a high level of protection to a low area, but generally no the other way around. Aircraft nav system and IFE.

My view of the contributions toward security that are inherent in the overall approach

I = Integrity
A = Availability
C = Confidentiality

Contribution to security continued…

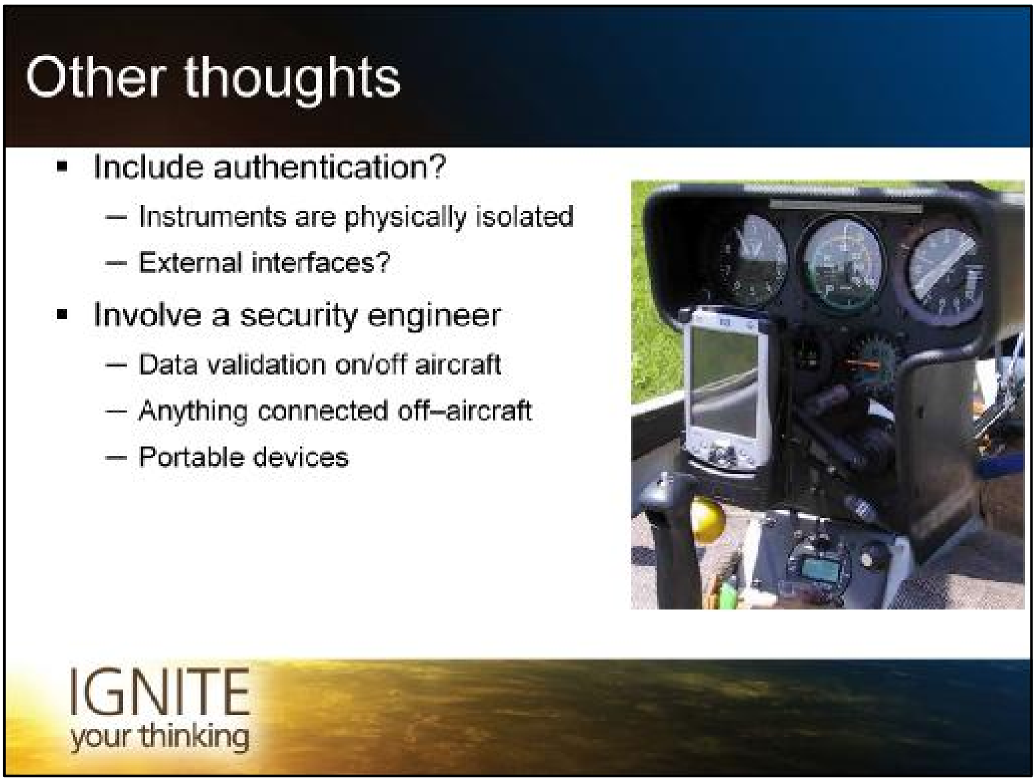
- Testing
  - Requirements coverage
  - Verified removal unused code / unreachable code
  - Structural coverage
  - Exercise all code MCDC
- QA - traceability, traceability, traceability
- Other factors
  - Physical isolation from external interfaces (tamper resistant?)
  - Physically isolated making replacing the software very difficult
  - Interface robustness testing (sunny day / rainy day)

IGNITE
your thinking

Contribution to security

- High barrier to entry:
  - Dev environments use actual aircraft hardware
  - Relatively uncommon OS's / hardware
- Small threat surface
  - Safety critical systems isolated from unauthorised personnel
  - Aviation specific data busses
- Other factors
  - Data chain of evidence (RTCA DO-200A)

IGNITE
your thinking



Pillars of security

https://upload.wikimedia.org/wikipedia/commons/e/ef/Glidercockpit.JPG - Wikimedia commons

Other thoughts – CC and Orange Book

- DO-178 - good foundation for Common Criteria trusted computing (EAL6+)
- Security requirements can be input as functional and technical requirements at the system level.
- Telemetry of flight controls (RPAS) is an excellent candidate for 'more security' (i.e. orange book)

IGNITE
your thinking

In relation to the Common Criteria requirements – under ISO 15408

- DO-178 lacks vulnerability assessment
- Though should be considered at the system safety level first
- Provides a very good foundation for Common Criteria

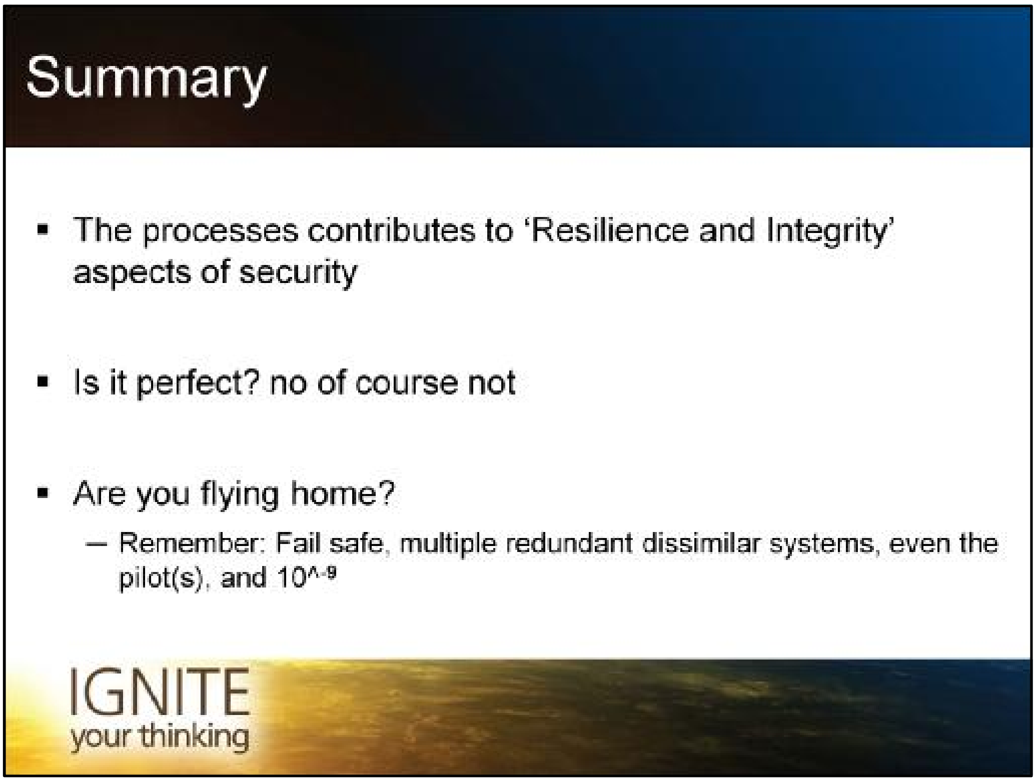
- Assurance levels
- EAL1: Functionally Tested
- EAL2: Structurally Tested
- EAL3: Methodically Tested and Checked
- EAL4: Methodically Designed, Tested and Reviewed
- EAL5: Semiformally Designed and Tested
- EAL6: Semiformally Verified Design and Tested
- EAL7: Formally Verified Design and Tested

Greenhill's integrity achieved NSA: EAL 6+ High Robustness Common Criteria SKPP—the highest security level ever achieved for an operating system (INTEGRITY-178 RTOS)

http://www.ghs.com/products/rtos/integrity.html
http://www.ghs.com/security/security_home.html

http://www.cotsjournalonline.com/articles/view/100490 JOE WLAD 2006
DO-178B and the Common Criteria: Future Security Levels Although there are similarities between the airborne safety-critical requirements in RTCA/DO-178B and the Common Criteria, ISO 14508, compliance with the higher levels of security in the Common Criteria demands meeting additional security requirements.

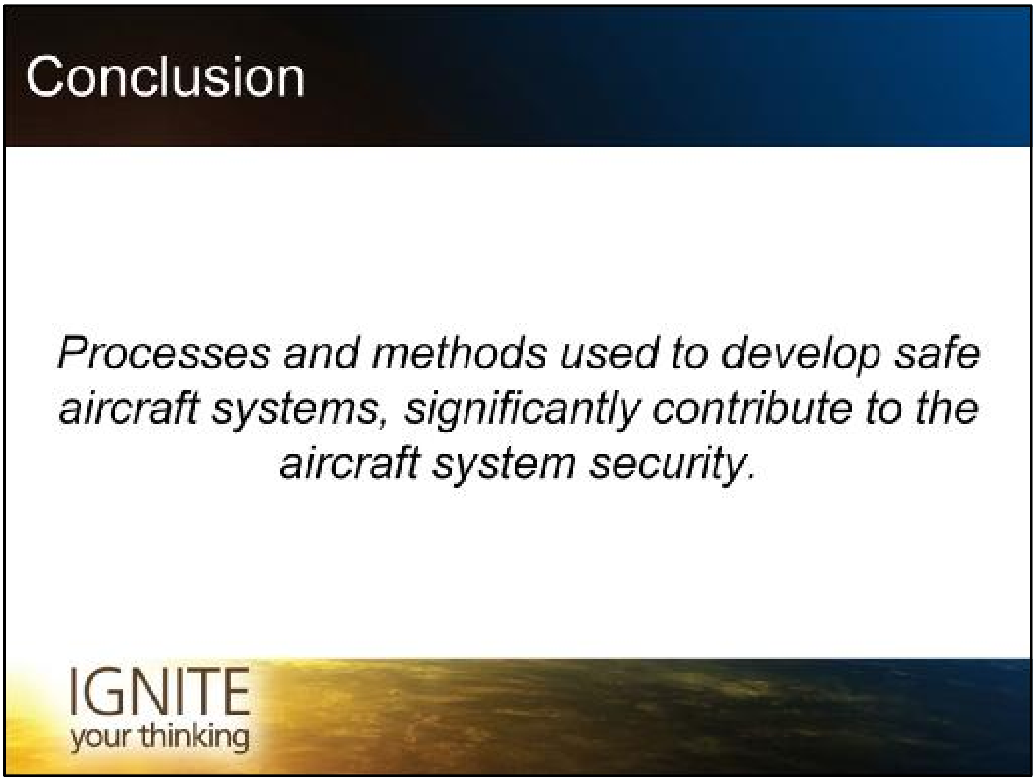
https://upload.wikimedia.org/wikipedia/commons/d/d8/CSIRO_ScienceImage_10876_Camclone_T21_Unmanned_Autonomous_Vehicle_UAV_fitted_with_CSIRO_guidance_system.jpg - Wikimedia Commons



Availability
The systems responsible for delivering, storing and processing information are accessible to authorized users when required.

Integrity
Information is accurate, authentic, complete and reliable.

Confidentiality
Information is disclosed only to authorized persons or organizations.



The processes and methods used to develop safe aircraft significantly contribute to aircraft system security… from external attack.

References

- *Avionics Development and Implementation, Cary R. Spitzer*
- *Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance, Leanna Rierson.*
- *DO-178B and the Common Criteria: Future Security Levels* - Joe Wlad
- *DO-178C Software Considerations in Airborne Systems and Equipment Certification – RTCA*
- *DO-178 Project development - Time (development and correctness phases) – Vance Hilderman*

IGNITE
your thinking

igniteyourthinking.beca.com

IGNITE
your thinking