CPP-Summit 2019

全球C++软件技术大会

C++ Development Technology Summit

Boolan
高端IT互联网教育平台

What Air Disasters Tell Us About Safety Critical Designs

MATTHEW BUTLER

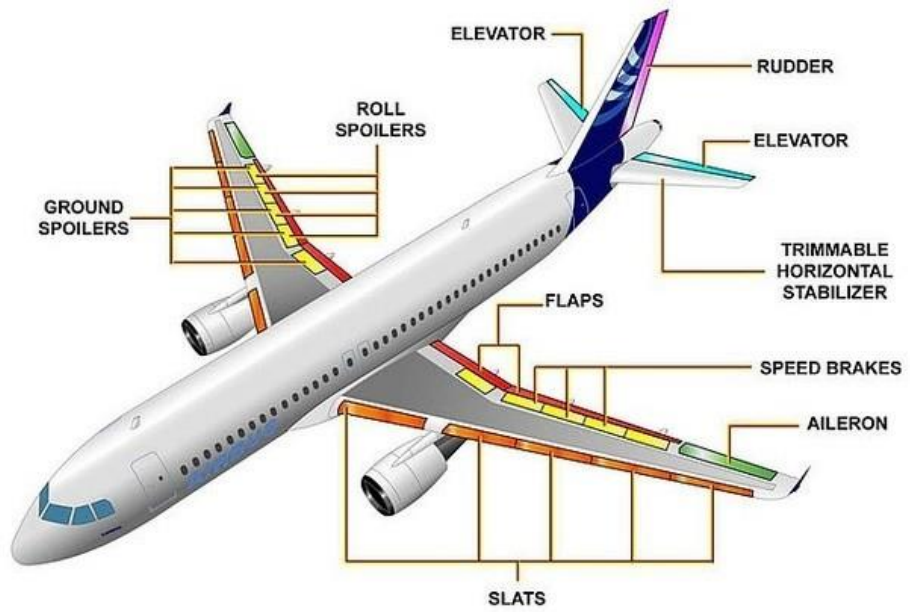
CPP-SUMMIT • NOVEMBER 2, 2019

About Me

- Software architect and security researcher
  - Started with C++ professionally in 1990 (Borland C++ 1.0)
  - C++Now and CppCon staff
  - International conference speaker and trainer
- Areas of expertise:
  - Network and applications security
  - Safety critical systems
  - Real-time data analysis
  - Embedded systems
- Member of the ISO C++ Standards Committee
  - Evolution Working Group (EWG)
  - SG12 – Software vulnerabilities and safety critical systems
  - SG14 – Low Latency, embedded
  - SG21 – Contracts

The Basics Of Fixed Wing Flight

- Four fundamental forces
  - Thrust
  - Drag
  - Gravity
  - Lift



- Three primary flight control surfaces
  - Rudder (yaw)
  - Elevators (pitch)
  - Ailerons (roll)

Air Crash Investigations

- Where investigators begin
  - Four corners of the aircraft
  - TFOAs
  - The Flight Data Recorder and Cockpit Voice Recorder

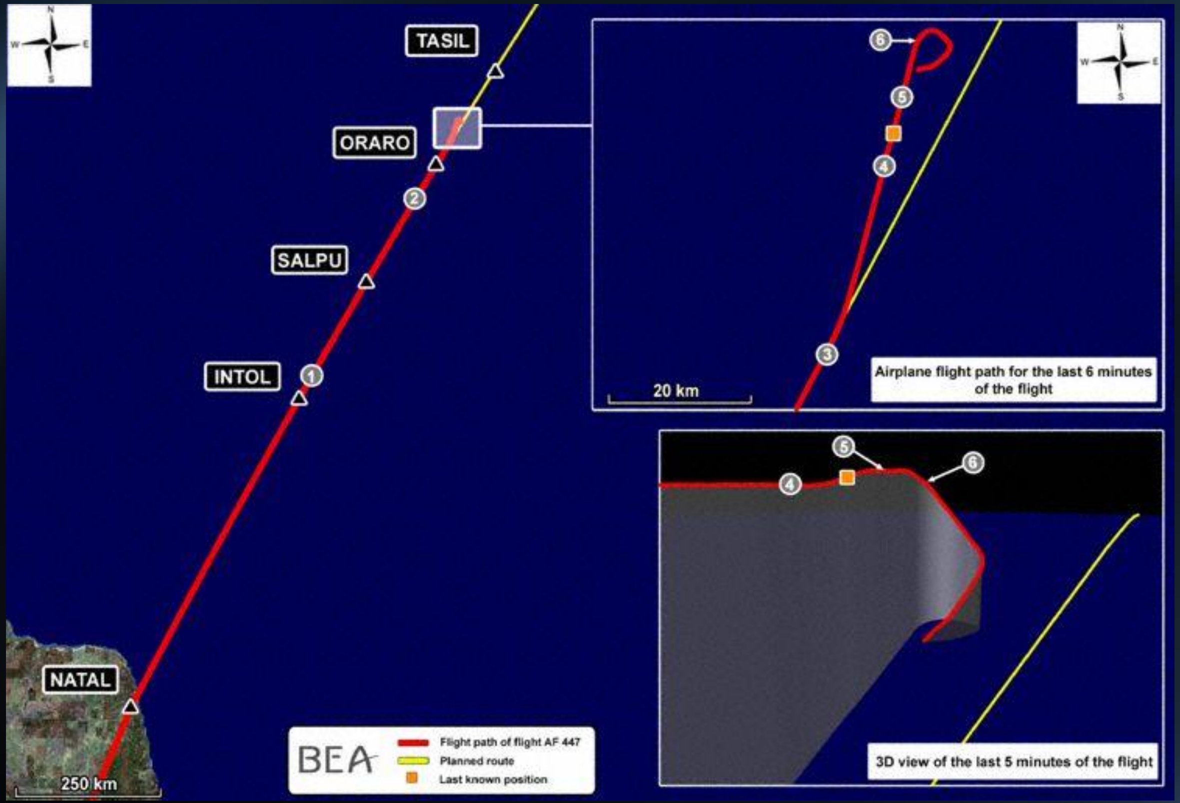
- What they investigate
  - Mechanical & Electrical
  - Aircraft history
  - Pilot training
  - Human factors
  - Weather

Air France 447

Air France 447

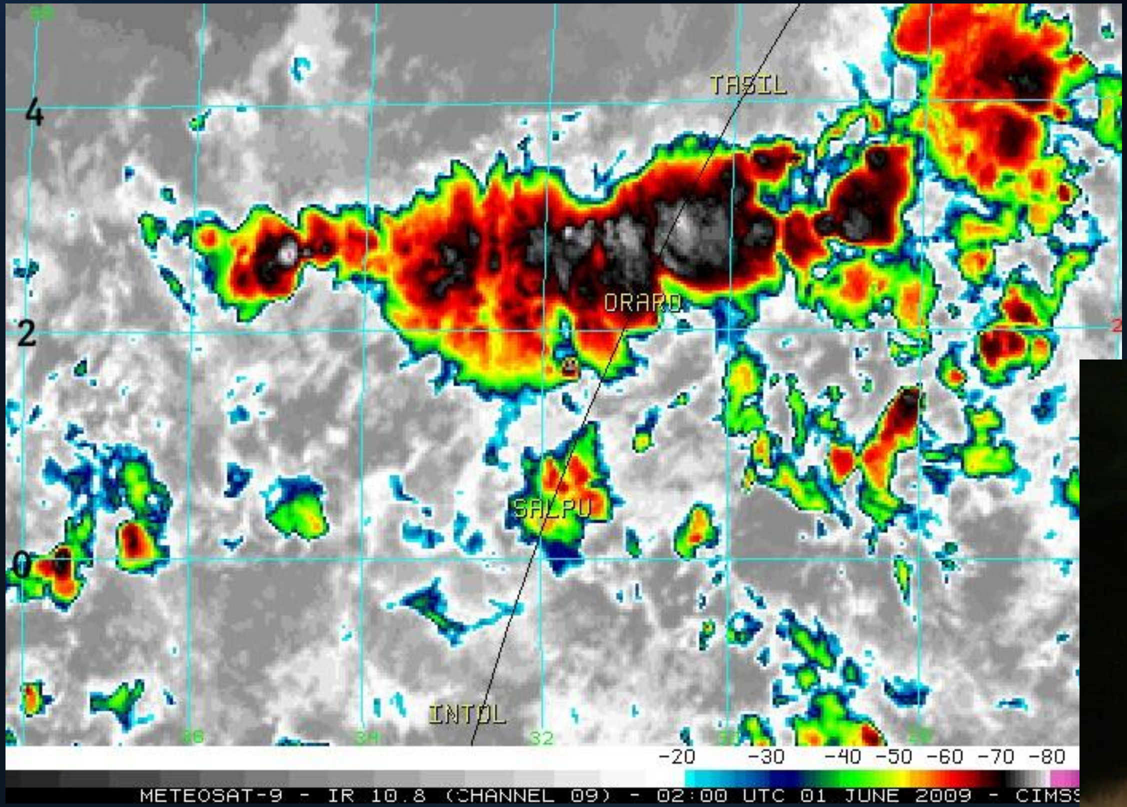
- Overnight flight from Rio de Janeiro to Paris on June 1, 2009
- Airbus A332 with three pilots
- Disappeared over the equatorial mid-Atlantic in a radar dead zone
- The flight data recorder and cockpit voice recorder would be found among the wreckage two years later
- All 228 passengers and crew on board were lost

Air France 447



What Is This Phenomena?

Air France 447



Air France 447

- During a storm, at night, with the captain on his rest period
- Super-cooled water froze on all 3 pitot tubes
- Starved of air speed data the autopilot shut down, auto-thrust shut down…
- Emergent behavior of complex systems
- Pilot error – the pilots stalled the jet down to the ocean surface

- Time from emergency to impact: 4 minutes

User Interfaces

- "Interfaces should be easy to use correctly and hard to use incorrectly" – Scott Meyers
- Design interfaces for task saturated, highly stressed pilots
- Assume non-optimal conditions
- Synchronize tasking options
- De-clutter human interfaces
- Never surprise the pilots

Air France 447

2:10:10 – AUTO FLT AP OFF

2:10:16 – AUTO FLT REAC W/S DET FAULT

2:10:23 – F/CTL ALTN LAW

2:10:29 – FLAG ON CAPT PFD SPD LIMIT

2:10:41 – FLAG ON F/O PFD SPD LIMIT

2:10:47 – AUTO FLT A/THR OFF

2:10:54 – NAV TCAS FAULT

2:11:00 – FLAG ON CAPT PFD FD

2:11:15 – FLAG ON F/O PFD FD

2:11:21 – F/CTL RUD TRV LIM FAULT

2:11:27 – MAINTENANCE STATUS EFCS 2

2:11:42 – MAINTENANCE STATUS EFCS 1

2:11:49 – EFCS2 1,EFCS1,AFS,,,,,PROBE-PITOT 1X2 / 2X3 / 1X3 (9DA),HARD

2:11:55 – EFCS1 X2,EFCS2X,,,,,,FCPC2 (2CE2) /WRG:ADIRU1 BUS ADR1-2 TO FCPC2,HARD

2:12:10 – FLAG ON CAPT PFD FPV

2:12:16 – FLAG ON F/O PFD FPV

2:12:51 – NAV ADR DISAGREE

2:13:08 – ISIS 1,,,,,,,ISIS(22FN-10FC) SPEED OR MACH FUNCTION,HARD

2:13:14 – IR2 1,EFCS1X,IR1,IR3,,,,ADIRU2 (1FP2),HARD

2:13:45 – F/CTL PRIM 1 FAULT

2:13:51 – F/CTL SEC 1 FAULT

2:14:14 – MAINTENANCE STATUS ADR 2

2:14:20 – AFS 1,,,,,,,FMGEC1(1CA1),INTERMITTENT

2:14:26 – ADVISORY CABIN VERTICAL SPEED

Air France 447

2:10:10 – **AUTO PILOT OFF**

2:10:16 – AUTO FLT REAC W/S DET FAULT

2:10:23 – **FLIGHT PROTECTION OFF**

2:10:29 – FLAG ON CAPT PFD SPD LIMIT

2:10:41 – FLAG ON F/O PFD SPD LIMIT

2:10:47 – **AUTO THROTTLE OFF**

2:10:54 – **TCAS OFF**

2:11:00 – FLAG ON CAPT PFD FD

2:11:15 – FLAG ON F/O PFD FD

2:11:21 – F/CTL RUD TRV LIM FAULT

2:11:27 – MAINTENANCE STATUS EFCS 2

2:11:42 – MAINTENANCE STATUS EFCS 1

2:11:49 – EFCS2 1,EFCS1,AFS,,,,,PROBE-PITOT 1X2 / 2X3 / 1X3 (9DA),HARD

2:11:55 – EFCS1 X2,EFCS2X,,,,,,FCPC2 (2CE2) /WRG:ADIRU1 BUS ADR1-2 TO FCPC2,HARD

2:12:10 – FLAG ON CAPT PFD FPV

2:12:16 – FLAG ON F/O PFD FPV

2:12:51 – NAV ADR DISAGREE

2:13:08 – ISIS 1,,,,,,,ISIS(22FN-10FC) SPEED OR MACH FUNCTION,HARD

2:13:14 – IR2 1,EFCS1X,IR1,IR3,,,,ADIRU2 (1FP2),HARD

2:13:45 – **PRIMARY NAVIGATION OFF**

2:13:51 – **SECONDARY NAVIGATION OFF**

2:14:14 – MAINTENANCE STATUS ADR 2

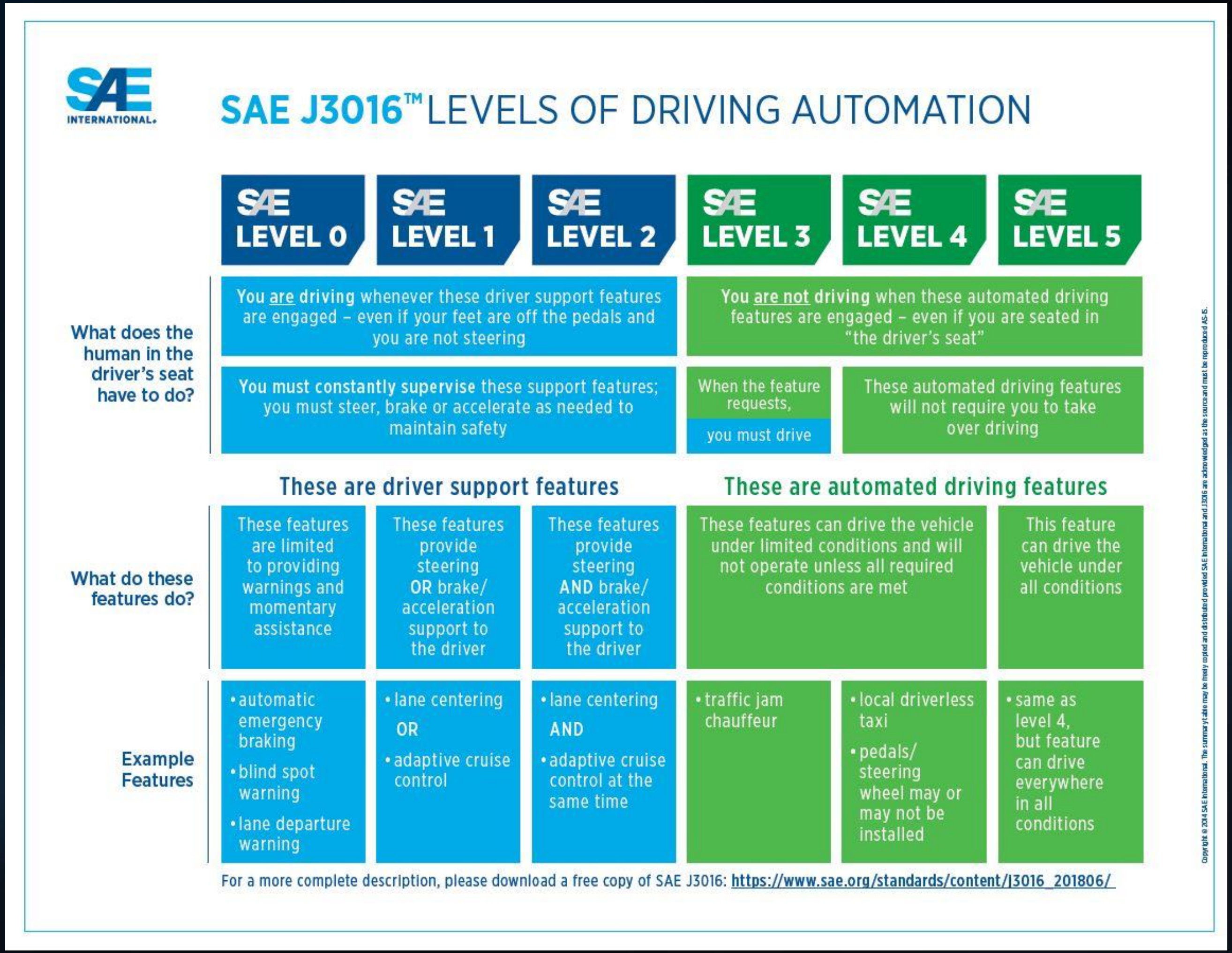
2:14:20 – AFS 1,,,,,,,FMGEC1(1CA1),INTERMITTENT

2:14:26 – **VERTICAL SPEED**

Error Messages

- "Once the wheels leave the deck, you automatically lose half your brain stem power" – F-18 Pilot
- Make error messages understandable for task saturated, highly stressed pilots
- There is a fine line between terse and unusable
- All messages should point to a solution
- Not all error messages matter at any given time

Automation


SAE INTERNATIONAL
SAE J3016™ LEVELS OF DRIVING AUTOMATION
SAE LEVEL 0
SAE LEVEL 1
SAE LEVEL 2
SAE LEVEL 3
SAE LEVEL 4
SAE LEVEL 5
What does the human in the driver's seat have to do?
You are driving whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering
You are not driving when these automated driving features are engaged – even if you are seated in "the driver's seat"
You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety
When the feature requests,
you must drive
These automated driving features will not require you to take over driving
These are driver support features
These are automated driving features
What do these features do?
These features are limited to providing warnings and momentary assistance
These features provide steering OR brake/ acceleration support to the driver
These features provide steering AND brake/ acceleration support to the driver
These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met
This feature can drive the vehicle under all conditions
Example Features
• automatic emergency braking
• blind spot warning
• lane departure warning
• lane centering
OR
• adaptive cruise control
• lane centering
AND
• adaptive cruise control at the same time
• traffic jam chauffeur
• local driverless taxi
• pedals/ steering wheel may or may not be installed
• same as level 4, but feature can drive everywhere in all conditions
For a more complete description, please download a free copy of SAE J3016: https://www.sae.org/standards/content/J3016_201806/

Automation

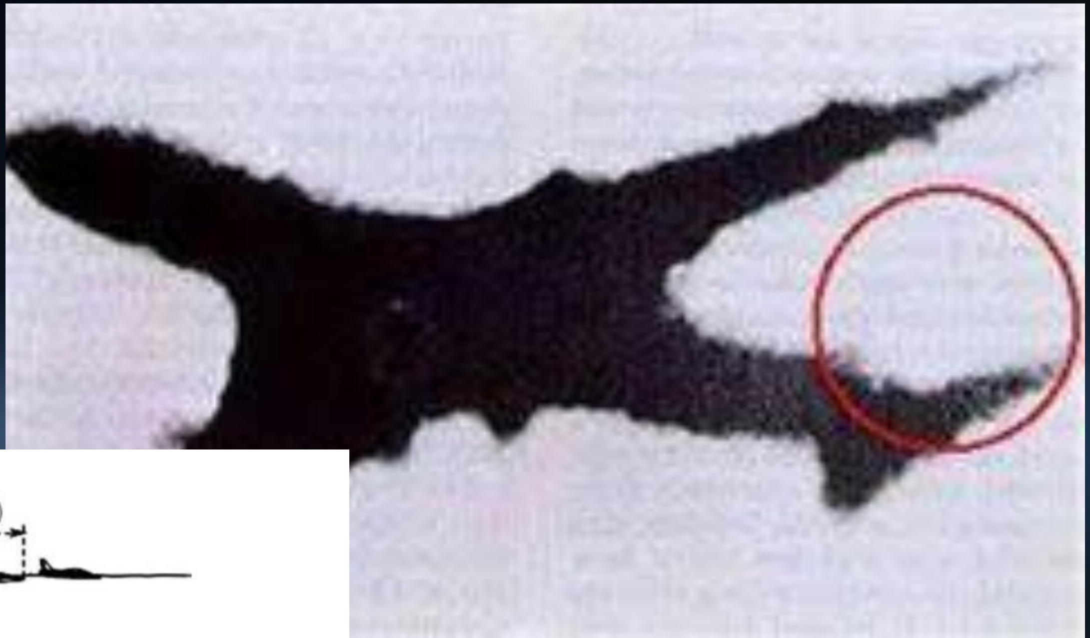
- Don't assume that pilots "fly" the aircraft
- Operating any kind of safety critical system is a perishable skill
- Design for the least experienced pilots
- Never surprise the pilots
- Plan for automation failures
- What is the default action for an automation failure?

Japan Airlines 123

Japan Airlines 123

- Scheduled flight from Tokyo’s Haneda Airport to Osaka International Airport on August 12, 1985
- Boeing 747SR
- Sudden decompression twelve minutes into flight
- Put the aircraft into a phugoid cycle
- Crashed into Osutaka Ridge 32 minutes later
- 520 dead, 4 survivors

Japan Airlines 123



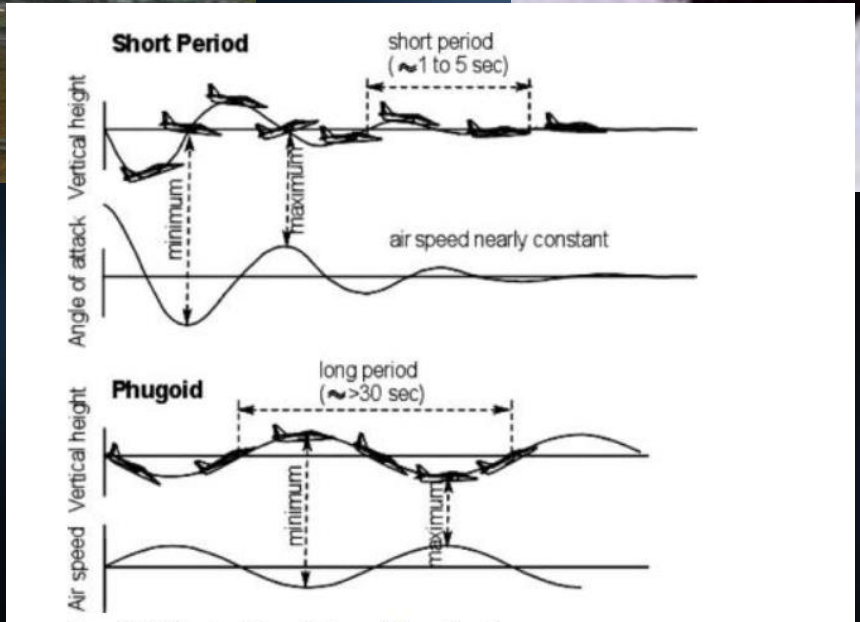


Figure 12.12. Short-period oscillations and phugoid motion

Japan Airlines 123

- The aircraft suffered a tail strike 7 years earlier
- Boeing performed the repair on the pressure dome incorrectly
- Hidden defects come out under stress
- The patch failed rupturing the pressure dome, blowing off the rudder and severing all the hydraulic lines
- Bad design – all the hydraulic lines meet in the tail with no shut off values

Japan Airlines 123

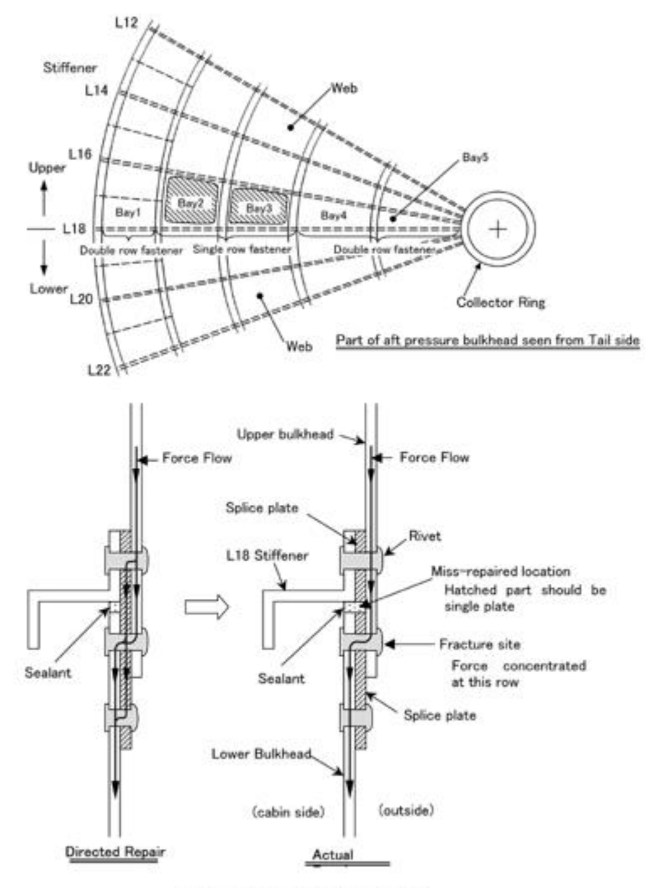
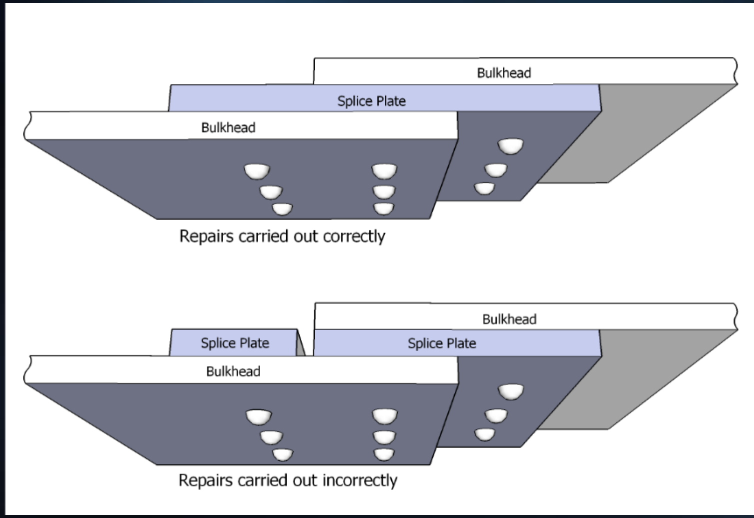


Fig.5 The Aspect of Aft Bulkhead Repair



Fail Safe Designs

- Fail safe designs are planned failures
- Test for modes of failure
- Every failure mode should have at least one backup
- Practice defense in depth
- Actively design to eliminate single points of failure

Single Points Of Failure

- Single points of failure are not always about mechanical systems
- Synchronization, storage, communications channels all present single points of failure
- What is your plan for mitigating single points of failures?
- What is the one single point of failure you can't mitigate?
- Auditing becomes critical to understanding failure

Fix It Right The First Time

- Technical debt is the price you pay for cutting corners
- Practice simplicity of design
- Complexity creates emergent behavior
- No design is ever self-documenting
- “Perfect Is The Enemy Of Good”
- When it comes to safety critical designs, “Good Is The Enemy Of Safe”

Boeing 737 MAX

Boeing 737 MAX

- Two crashes of Boeing 737 MAX aircraft 5 months apart
  - Lion Air accident: October 29, 2018
  - Ethiopian Airlines accident: March 10, 2019
- Initially blamed on pilot error
- The aircraft design came under suspicion causing all MAX aircraft to be grounded world-wide
- 346 dead

Boeing 737 MAX

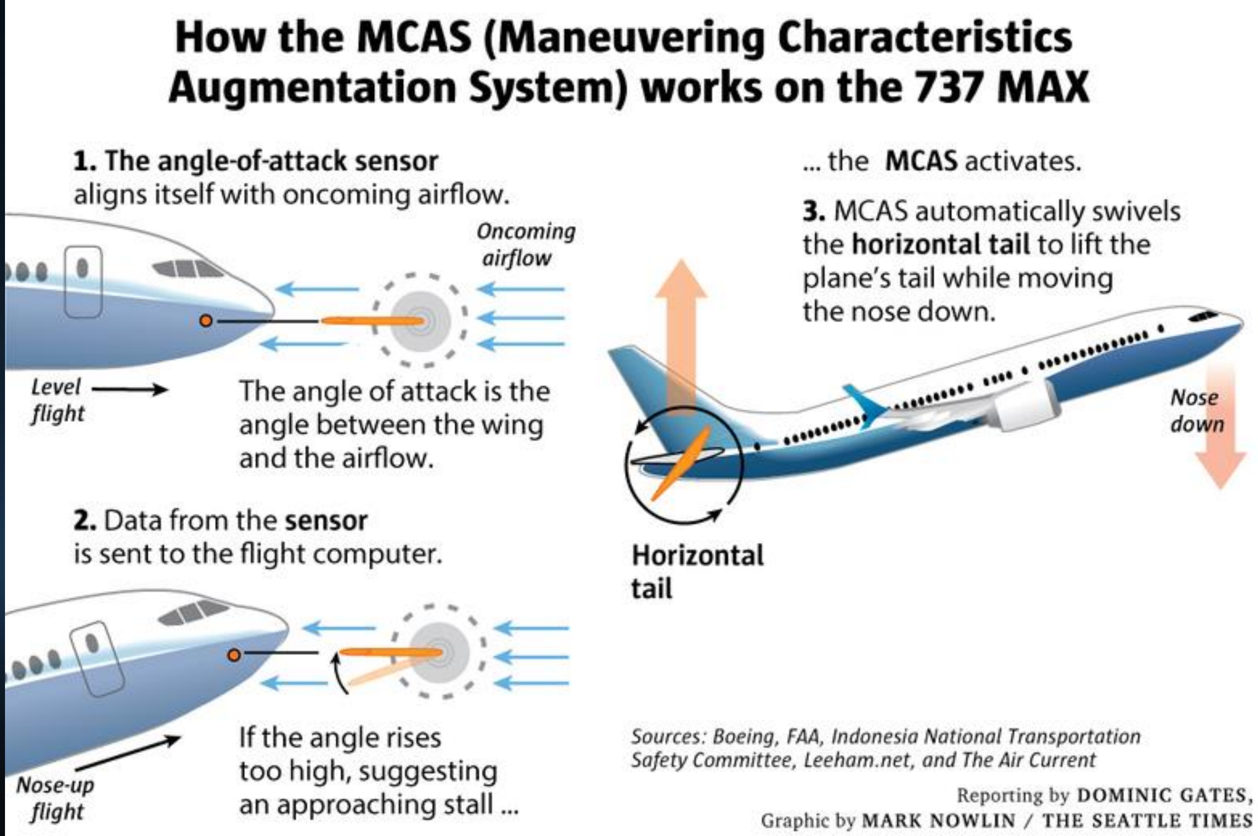
- Facing pressure from Airbus, Boeing modified the 737 airframe
- Needed to handle larger, more fuel-efficient engines
- The under carriage of the 737 is too low for larger engines
- Promised customers that their pilots would not need simulator time on the new aircraft
- Delivered the upgrade documentation on a tablet
- Forgot one important “feature”

Boeing 737 MAX

Boeing 737 MAX

- This forced a re-design of the engine nacelles, raising them and pushing them forward
- The new configuration caused an imbalance in the flight characteristics
- 737 MAX had the habit of pitching up inducing a stall
- So they added the Maneuvering Characteristics Augmentation System (MCAS) system to push the nose back down
- MCAS was not considered a safety of flight risk
- Boeing was allowed to self certify but knew MCAS had problems

Boeing 737 MAX



Boeing 737 MAX

- Boeing tried to make an ancient design work with a few mods
- MCAS system had only one Angle-of-Attack (AOA) sensor
- Created a single point of failure
- Pilots were not trained on the new system or told it existed
- Silent systems create emergent behavior
- A software system to fix a hardware problem

Safety In The Name Of Economics

- Every company "advertises" themselves as safe, secure
- Safety and economics are always in contention
- Treat design flaws as design flaws
- Don't "patch" architectural problems
- It takes courage to admit that the design is inadequate to the job
- Avoid creating a system that fosters an over dependency on automation

Complexity Is The Enemy

- Complexity breeds emergent behavior
- Vulnerabilities always migrate to complexity
- Make behavior obvious (especially bad behavior)
- Practice simplicity
- Document complexity simply
- Never surprise the pilots

Managing Risk

- There are always risk trade-offs
- No system is 100% safe – the world doesn't work that way
- It's the difference between taking risks and taking stupid risks
- Safety critical systems have a special responsibility
- Ask yourself, "would I fly in/ride on/use this…"

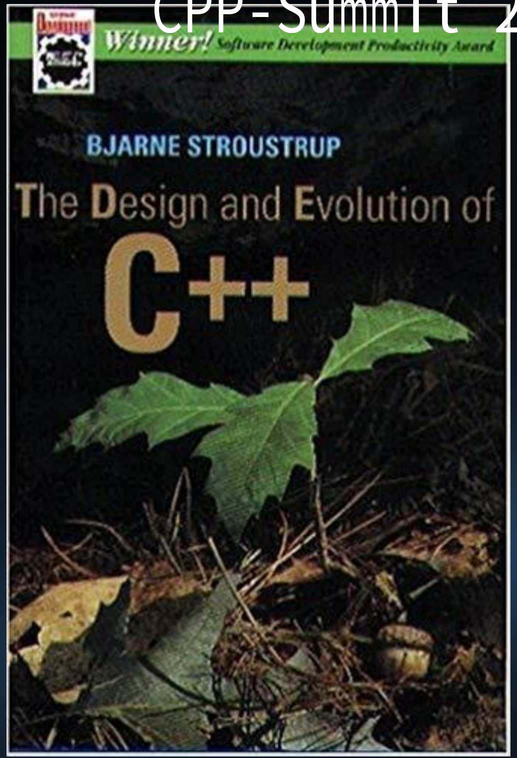
- Sometimes you will be asked to do the unthinkable…
- There is never a right reason to do the wrong thing

Final Thoughts

- When in doubt, always give the pilots the last word
- Always give the pilots the tools they need to solve the problem
- Automation is a silent partner – but it is only a partner
- Plan for failure
- Remember Occam's Razor
  - "All Other Things Being Equal, The Simplest Answer Is Usually The Right One"
- Never surprise the pilots

C++ History

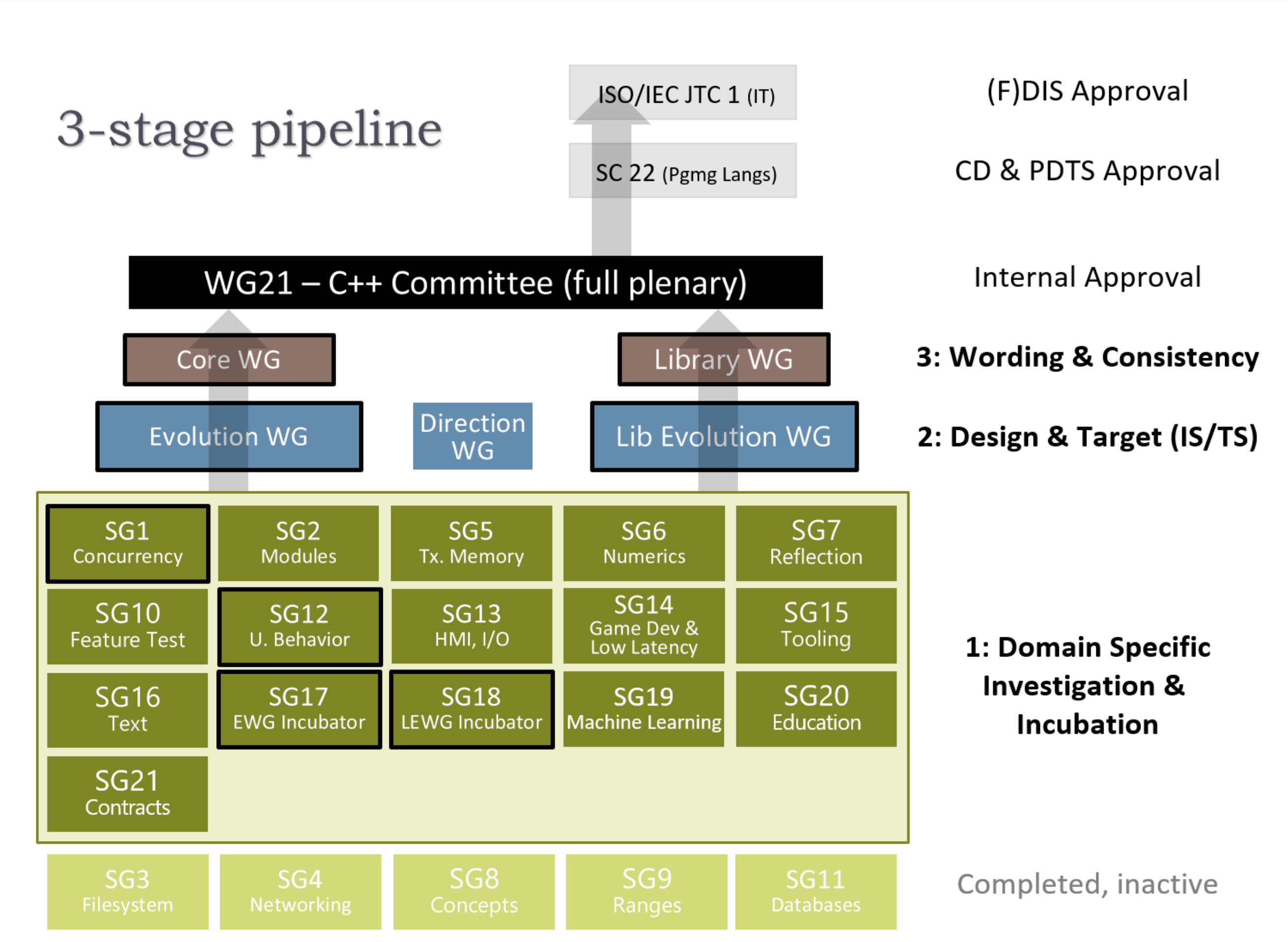
- Bjarne Stroustrup (creator)
  - 1979 – Started C with classes at Bell Labs
  - 1982 – Evolved from C with classes into C++
  - 1985 – Published "The C++ Programming Language"



- Releases
  - 1985 – First implementation of C++ released
  - 1989 – C++ 2.0 released
  - 1998 – C++98 released as the first standard
  - 2003 – C++03 released as a minor update
  - 2011 – C++11 (Modern C++): lambdas, std::move, RVO, constexpr, initializer lists, ++
  - 2014 – C++14: mostly bug fixes but started the regular release cadence
  - 2017 – C++17: Major revision to the standard
  - 2020 – Modules, Concepts, Ranges, Coroutines, spaceship operator, ++

The ISO C++ Standards Committee

- WG21
  - Herb Sutter, chair
  - International body
  - 3-year release cadence
  - 1500-page standard
    - Core language – 500
    - STL – 1000
  - 2 pipelines
    - Core language WG
    - Library WG
  - C++20 will be voted on in Prague (Feb 2020)



The Committee Over Time

First X3J16 meeting

Somerset, NJ, USA (1990)

Completed C++11

Madrid, Spain (2011)

Completed C++14

Issaquah, WA, USA (2014)

Photo: Chandler Carruth and Olivier Giroux. License: tinyurl.com/9wn439f

Completed C++17

Kona, HI, USA (2017)

C++20 And Beyond

Coming In C++20

- Concepts
- Ranges
- Coroutines
- std::atomic<std::shared_ptr<T>>
- std::format replaces printf()
- Modules
- constexpr++
- using enum
- Volatile deprecation
- Designated Initializers
- Spaceship operator (<=>)
- Calendar + time zone extensions
- std::span

Beyond C++20

- Contracts (pushed to C++23 at least)
- secure_clear (probably C++23)
- Zero-overhead Deterministic Exceptions (maybe C++23 but more likely C++26)
- Enumerating Core Undefined Behavior (great idea but probably won't make into the standard)
- More focus on safety critical and security issues