What FM can offer DFCS Design

John Rushby

Computer Science Laboratory
SRI International

Overview

- What has actually gone wrong in practice?
- What is the pattern?
- What is the solution?

Advanced Fighter Technology Integration (AFTI) F16

- Triplex DFCS to provide two-fail operative design
- Analog backup
- Digital computers were not synchronized
- "General Dynamics believed synchronization would introduce a single-point failure caused by EMI and lightning effects"

AFTI F16 DFCS Redundancy Management

- Each computer samples sensors independently, uses average of the good channels, with wide threshold
- Single output channel selected from among the good channels
- Output threshold 15% plus rate of change
- Four bad values in a row and the channel is voted out

AFTI F16 Flight Test, Flight 15

- Stores Management System (SMS) relays pilot requests for mode changes to DFCS
- An unknown failure in the SMS caused it to request mode changes 50 times a second
- DFCS responded at a rate of 5 mode changes per second
- Pilot said aircraft felt like it was in turbulence
- Analysis showed that if aircraft had been maneuvering at the time, DFCS would have failed

AFTI F16 Flight Test, Flight 36

- Control law problem led to "departure" of three seconds duration
- Sideslip exceeded 20°, normal acceleration exceeded −4g, then +7g, angle of attack went to −10°, then +20°, aircraft rolled 360°, vertical tail exceeded design load, failure indications from canard hydraulics, and air data sensor
- Side air data probe blanked by canard at high AOA
- Wide threshold passed error, different channels took different paths through control laws
- Analysis showed this would cause complete failure of DFCS and reversion to analog backup for several areas of flight envelope

AFTI F16 Flight Test, Flight 44

- Asynchronous operation, skew, and sensor noise led each channel to declare the others failed
- Analog backup not selected (simultaneous failure of two channels not anticipated)
- Aircraft flown home on a single digital channel
- No hardware failures had occurred

AFTI F16 Flight Test

- Repeated channel failure indication in flight was traced to roll-axis software switch
- Sensor noise and asynchronous operation caused one channel to take a different path through the control laws
- Decided to vote the software switch
- Extensive simulation and testing performed
- Next flight, same problem still there
- Found that although switch value was voted, the unvoted value was used

X29 Flight Test

- Three sources of air data on X29A: nose and two side probes
- If value from nose is within threshold of both side probes, use nose probe value
- Threshold is large due to position errors in certain flight modes
- If nose probe failed to zero at low speed it would still be within threshold of correct readings
- Aircraft would become unstable and "depart"
- Caught in simulation but 162 flights had been at risk

HiMAT Flight Test

- Single failure in redundant uplink hardware
- Software detected this, and continued operation
- But would not allow the landing skids to be deployed
- Aircraft landed with skid retracted, sustained little damage
- Traced to timing change in the software that had survived extensive testing

Gripen Fight Test, Flight 6

- Unstable aircraft
- Triplex DFCS with Triplex analog backup
- Yaw oscillations observed on several flights
- Final flight had uncontrollable pitch oscillations
- Crashed on landing, broke left main gear, flipped
- Traced to control laws

Space

- Voyager computer clocks skipped 8 seconds at Jupiter due to high radiation levels (AW&ST Aug 7, 1989)

- So "continuous resynchronization" provided at Neptune

- Also, remember STS-1: "The bug heard round the world" (SEN Oct 1981)

FDIR and Crew Interface

- Imaginary crash scenario
- Broken fan blade on port engine
- Port vibration sensor saturates, limiter cuts in
- Vibration travels down wing, shakes starboard engine
- Starboard vibration sensor reports the attenuated vibration
- Only starboard vibration warning light comes on in cockpit
- Pilot shuts down the good engine, crashes short of runway
- Similar to British Midland 737 crash in 1989

Complexity and Integration

- "The FMS of the A320 'was still revealing software bugs until mid-January,' according to Gérard Guyot (Airbus test and development director). There was no particular type of bug in any particular function, he says. 'We just had a lot of flying to do in order to check it all out. Then suddenly it was working,' he says with a grin" (Flight International, 27 Feb 1989)

- The ATF hardware is ready to go, but cannot be flown because the software engineers "can't get all the 0's and 1's in the right order" (Northrop Engineer, 7 Aug, 1990)

Complexity and Integration

As of early 1988	A300	A310	A320
Put in service	1982	1983	1988
Number in service	16	149	3
Flight Hours	16,000	810,000	2,000
	Computers		
Autopilot	2 FCC	2 FCC	2 FMGC
Rudder	2 FAC	2 FAC	2 FAC
Autothrottle	1 TCC	1 or 2 TCC	
Slats and flaps		2 SFCC	2 SFCC
Elevator/aileron		2 EFCU	2 ELAC
Spoilers		2 FLC	3 SEC
Fuel management		2 CGCC	
Instruments		3 SGU	3 DMC
Brakes			2 BSCU
Engines		2 FADEC	2 FADEC

Analog, Mechanical Backups

- Do mechanical and analog backups reduce the requirement for ultra-reliability in DFCS?
- Not if the DFCS is providing stability augmentation or envelope protection
- Similar problem in ATC—potential to move traffic at higher rates than the backup can handle
- No FAA certification credit for mechanical rudder and trim-tab on A320

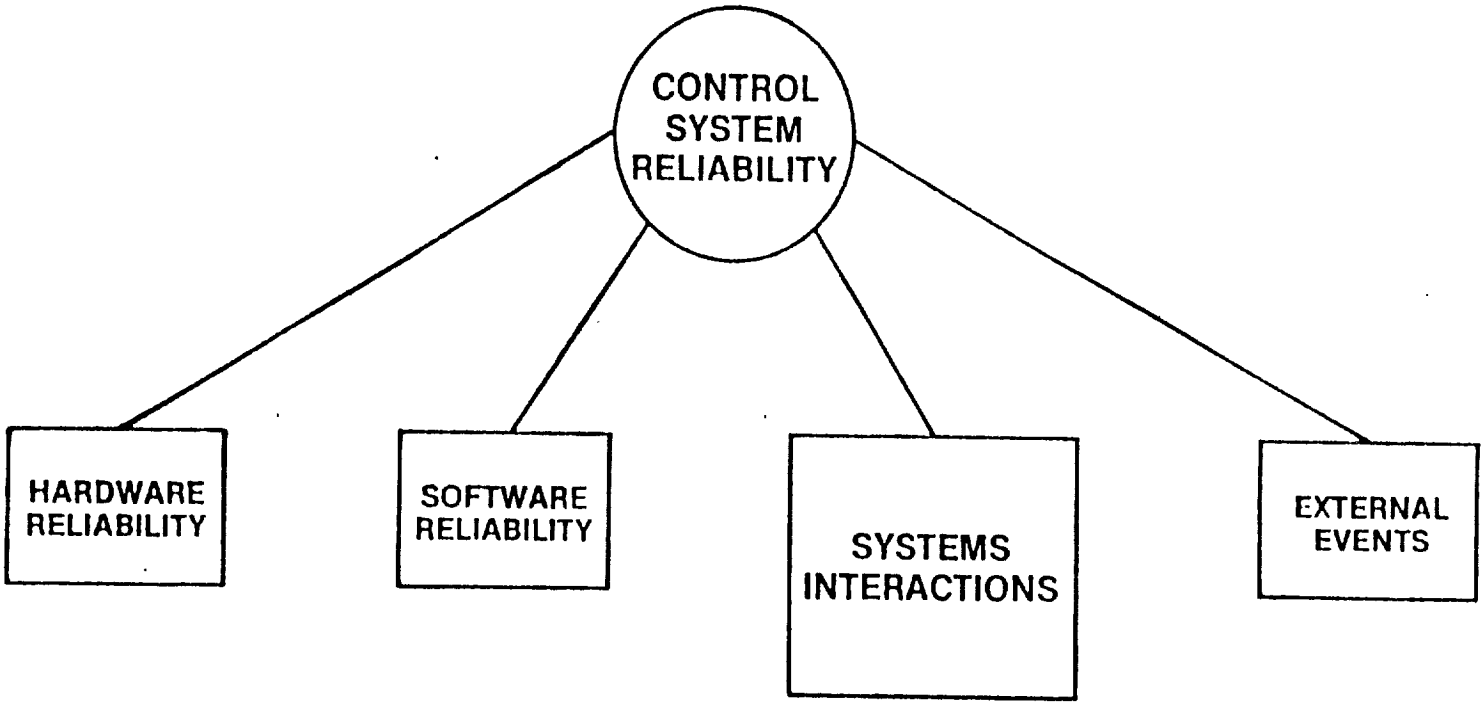
Analysis: Dale Mackall, NASA Engineer
AFTI F16 Flight Test

- Nearly all failure indications were not due to actual hardware failures, but to design oversights concerning asynchronous computer operation

- Failures due to lack of understanding of interactions among
  - Air data system
  - Redundancy management software
  - Flight control laws

FLIGHT CONTROL SYSTEM RELIABILITY HEAVILY DEPENDENT ON SYSTEM INTERACTIONS

NASA
DERI 83-099



Analysis: NASA-LaRC 1988 FCDS Technology Workshop

- Lack of fully effective design and validation methods with support tools to enable engineering of highly-integrated, flight-critical digital systems

- Complexity of failure containment, test coverage, FMEA, redundancy management, especially in the face of increased integration of flight-critical functions

- Sources of failure:
  - Multiple independent faults (never observed)
  - Single point failures (observed sometimes)
  - Domino failures (most common?)

Analysis: Scientific Foundations

- It is time to place the development of real-time systems on a firm scientific basis. Real-time systems are built one way or another because that was the way the last one was built. And, since the last one worked, we hope that the next one will. (Fred Schneider)

- "Not far from there (CNRS-LAAS), Airbus Industries builds the Airbus A320s. These are the first commercial aircraft controlled solely by a fault-tolerant, diverse computing system. *Strangely enough this development owes little to academia.* (IEEE Micro, April 1989, p6)

Analysis

- The problems of DFCS are the problems of systems whose complexity has exceeded the reach of the intellectual tools employed

- Intuition, experience, and techniques derived from mechanical and analog systems are insufficient for complex, integrated, digital systems

Synthesis

- Computer science has been addressing issues of systematic design, fault tolerance, and the mastery of complexity with some (limited) success for the last 20 years

- But there has been little interest in learning about, and applying this knowledge to, real-time control systems in general (and little opportunity to apply it to DFCS)

- And little of the lore and wisdom of practical real-time control system design has been captured and analyzed

What Computer Science Can Offer DFCS

- Systematic techniques for the construction of trustworthy software, including:
  - Techniques for the precise specification of requirements and the development of designs
  - Systematic approaches to the design and structuring of distributed and concurrent systems
  - Fault tolerant algorithms
  - Systematic methods of testing and analytic methods of verification
- Where do formal methods come in?

Applied Mathematics and Engineering

- Established engineering disciplines use applied mathematics
  - As a *notation* for describing systems
  - As an analytical tool for calculating and *predicting* the behavior of systems
- Computers can provide speed and accuracy for the calculations

Applied Mathematics and Software Engineering

- The applied mathematics of software is formal logic
- Formal Logic can provide
  - A notation for describing software designs—*formal specification*
  - A calculus for analyzing and predicting the behavior of systems—*formal verification*
- Computers can provide speed and accuracy for the calculations
- Calculating the behavior of software is an exercise in formal reasoning—i.e., theorem proving

Formal Methods

- Methodologies for using mathematics in software engineering

- Can be applied at many different levels, for both description and analysis

  0. No application of formal methods
  1. Quasi-formal pencil and paper techniques
  2. Mechanized quasi-formal methods
  3. Fully formal pencil and paper techniques
  4. Mechanically checked fully formal techniques

Benefits of Formal Specification

- Unambiguous description facilitates communication among engineers
- Early detection of certain errors
- Encourages systematic, thoughtful approach, reuse of well-understood concepts
- As documentation, reduces some of the difficulties in maintenance and modification

Benefits of Formal Verification

- Subjects the system to extreme scrutiny, increasing designers' understanding of their own creation
- Helps identify assumptions, increases confidence
- Encourages simple, direct designs, austere requirements—better systems
- Encourages and supports a systematic, derivational approach to system design
- Complements testing and allows it to focus on fundamentals

Conclusion: What FM Can Offer DFCS

- Precise *notations* for specifying requirements and designs
- *Concepts* and *structure* for systematic design
- Intellectual tools for *analyzing* the consistency of specifications and the conformance of designs
- A way to regain *intellectual mastery* of complex systems and their interactions

Recommendations

- Just adding formal methods to existing practice is inappropriate
- Capture and analyze lore and wisdom (and mistakes) of actual DFCS designs
- Apply modern Computer Science (including Formal Methods) to develop building blocks for principled DFCS design
- Ultimately, build one and fly it!