EDA122/DIT061 Fault-Tolerant Computer Systems
DAT270 Dependable Computer Systems

Welcome to Lecture 7

Generalized Stochastic Petri-Nets (GSPNs)
Software redundancy and Design Diversity
Airbus A330/A340 Fly-by-wire system

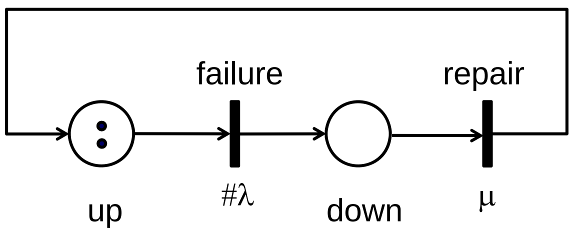
Outline

- Generalized Stochastic Petri Nets (GSPNs)
  - Availability GSPN model of hot standby systems
  - Reachability graph
  - Elements of GSPN:s
  - Examples: construction of GSPN models for various systems
- Software redundancy
  - Design diversity
  - N-version programming
  - Recovery blocks
  - Design diversity in Airbus A330/A340 fly-by-wire system

Generalized Stochastic Petri Nets (GSPN)

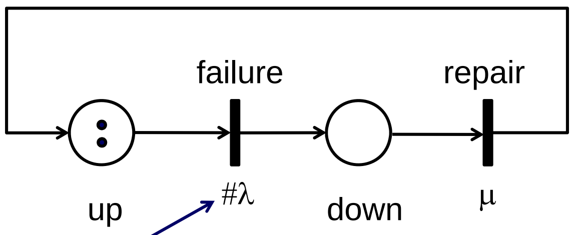
- A GSPN provides a graphical syntax for specifying state space models (Markov models)
- It provides a more compact way of describing a state space model than a state diagram
- A Petri net consists of
  - Places (circles)
  - Transitions (vertical bars)
  - Arcs (arrows)
  - Tokens (dots)

GSPN modell of repairable hot standby system with one spare units



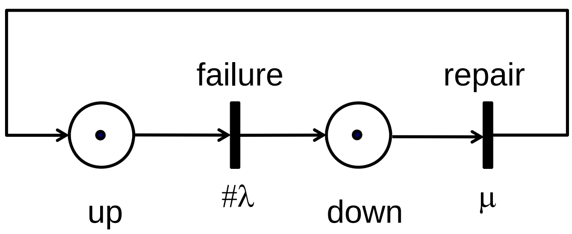
Marking shows the case when both modules are working: there are two tokens in the place “up”

GSPN modell of repairable hot standby system with one spare unit



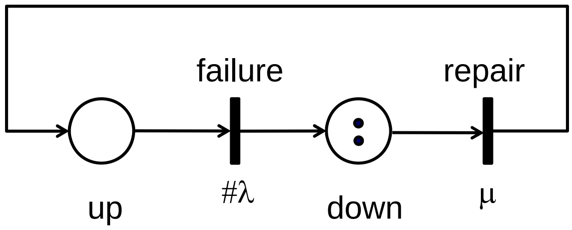
The # character indicates that the firing rate depends on the number of tokens in the place "up"

GSPN modell of repairable hot standby system with one spare unit



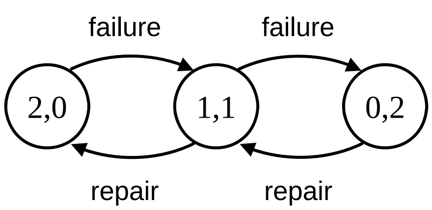
Marking when one module is up and one is down

GSPN modell of repairable hot standby system with one spare unit

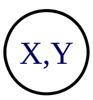


Marking when both modules are down

Reachability Graph for the GSPN model

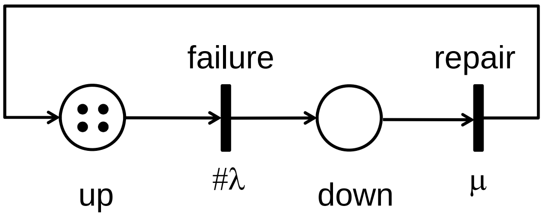


State labelling:



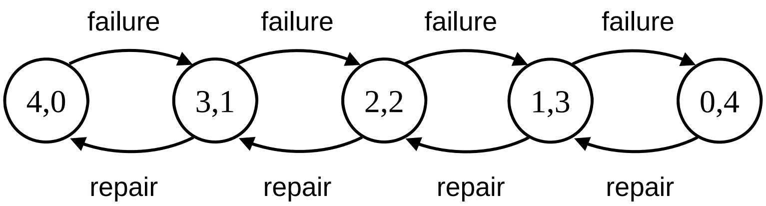
X = Number of tokens in “up”
Y= Number of tokens in “down”

GSPN modell of repairable hot standby system with 3 spare units

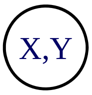


Marking when all modules are working (= four tokens in place "up")

Reachability Graph for hot standby system with 3 spares



State labelling:

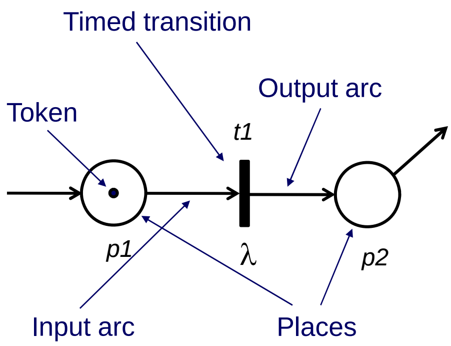


X = Number of tokens in "up"
Y= Number of tokens in "down"

Elements of GSPNs

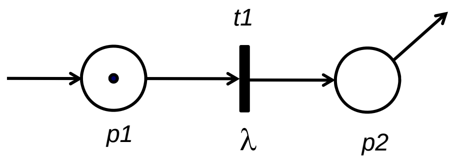
- Places – holds tokens
- Transitions – moves tokens from one place to another
- Arcs – connects transitions with places
- Tokens – moves between places via transitions

- Marking – a certain placement of tokens in the Petri net.

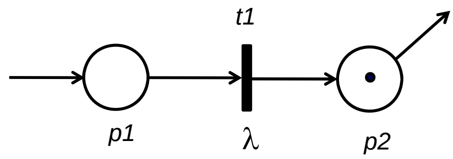


Timed Transitions in GSPNs

- Timed transitions are drawn as a *thick* vertical line
- The timed transition *t1* fires at a random point in time after a token has arrived in *p1*
- The firing time is exponentially distributed with the rate λ
- When *t1* fires, one token moves from *p1* to *p2*
- In this example, the firing rate is constant, i.e., independent of the number of tokens in *p1*.



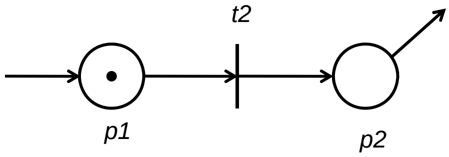
Marking before *t1* has fired



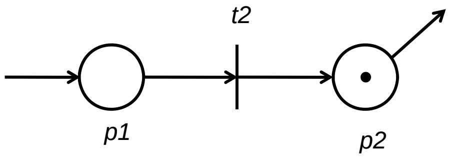
Marking after *t1* has fired

Immediate Transition in GSPNs

- An immediate transition is drawn as a *thin* vertical line.
- *t2* fires immediately when one token has arrived in *p1*



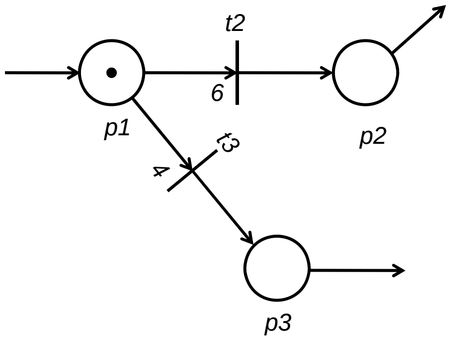
Before immediate transition fires



After immediate transition fires

Weights for Immediate Transitions

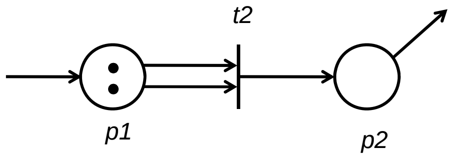
- Immediate transitions leaving the same states can be assigned weights
- In this example, *t2* has a weight of 6 and *t3* has a weight of 4, which means that *t2* fires with 60% probability and *t3* fires with 40% probability when a token enters *p1*
- Note: Weights are normalized to sum up to one when the GSPN is analysed



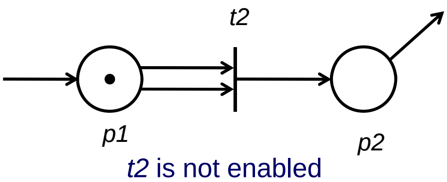
Competing immediate transitions with weights

Arc Multiplicity

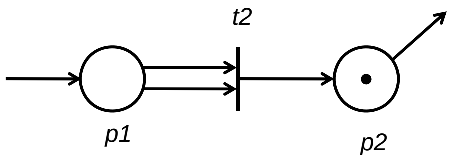
- Both timed and immediate transitions can have multiple input and output arcs.
- In this example, *t2* has two inputs arcs and thus fires when *p1* contains two tokens.



t2 is enabled and fires



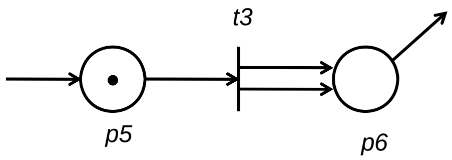
t2 is not enabled



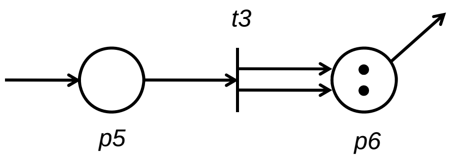
Marking after *t2* has fired

Multiple Output Arcs

- In this example, *t3* has two outputs arcs and thus produces two tokens when it fires.



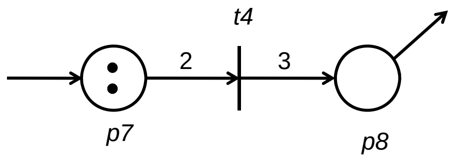
t3 is enabled and fires



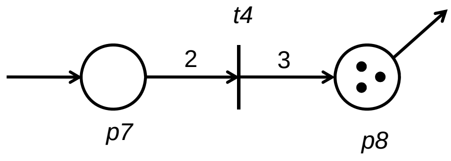
Marking after *t3* has fired

Simplified notation for multiple arcs

- The number of input and output arcs can be given by number placed just above an arc.
- In this example, *t4* has 2 input arcs and 3 output arcs.



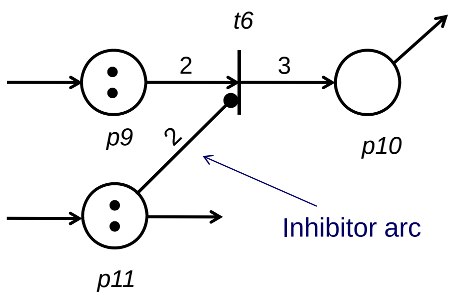
Before *t4* fires



After *t4* has fired

Inhibitor arcs

- An inhibitor arc blocks the firing of a transition based on the marking of a place
- If *p11* has 2 or more tokens the inhibitor arc blocks the firing of *t6*



Problems

1. Construct a GSPN model for calculating the reliability of a system consisting of two modules operating in active redundancy.
2. Construct a GSPN model for calculating the reliability of a TMR system
3. Construct a GSPN model for calculating the reliability of a *k-of-n* system

GSPN reliability model for system with two modules operating in active redundancy

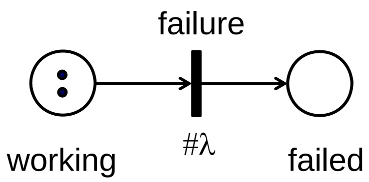
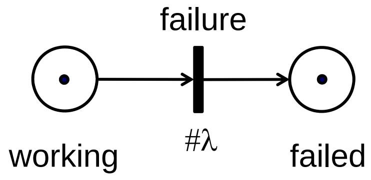


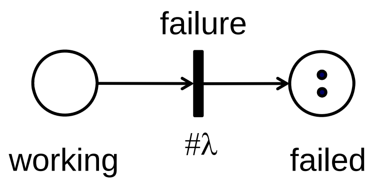
Figure shows GSPN model with initial marking, i.e., with two working modules

GSPN reliability model for system with two modules operating in active redundancy



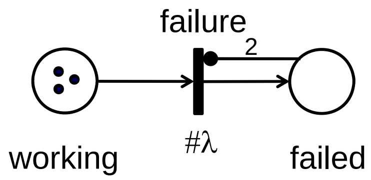
Marking corresponding to one working and one failed module

GSPN reliability model for system with two modules operating in active redundancy



Marking corresponding to system failure

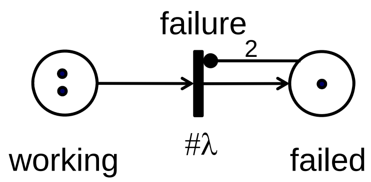
GSPN reliability model for TMR system



Marking corresponding to three modules working

Lecture 7 EDA122/DIT061 Fault-Tolerant Computer Systems / DAT270 Dependable Computer Systems 23

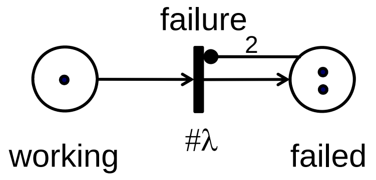
GSPN reliability model for TMR system



Marking corresponding to two modules working, one module failed

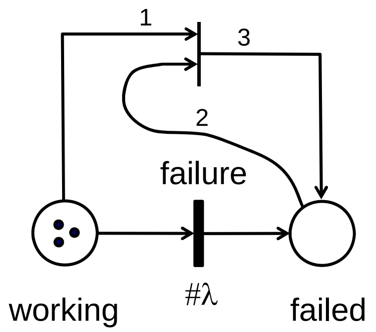
Lecture 7 EDA122/DIT061 Fault-Tolerant Computer Systems / DAT270 Dependable Computer Systems 24

GSPN reliability model for TMR system



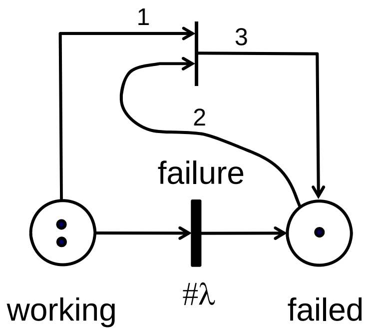
Marking corresponding to one module working, two modules failed
Timed transition is disabled by inhibitor arc

GSPN reliability model for TMR system



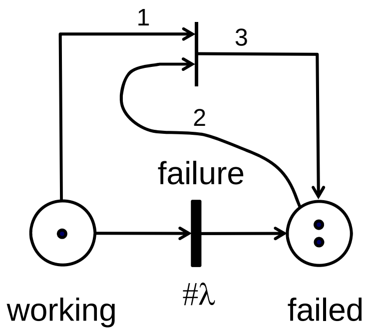
Three modules working

GSPN reliability model for TMR system



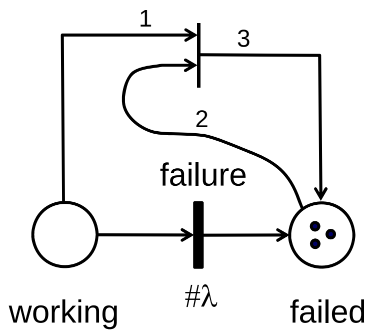
Two modules working, one module failed

GSPN reliability model for TMR system



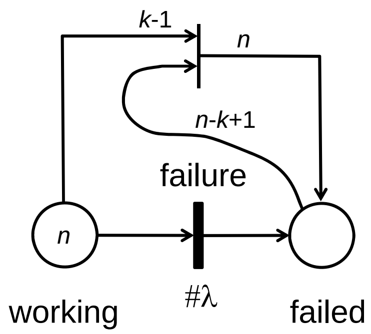
One module working, two modules failed => immediate transition enabled

GSPN reliability model for TMR system



Marking after immediate transition has fired – corresponds to system failure

GSPN reliability model for k-of-n system



Marking corresponding to *n* modules working

Software redundancy

Software redundancy techniques can be divided in two major classes:

- With diversity
  - Aim is to tolerate software development faults
  - Design diversity
  - Data diversity
- Without diversity
  - Aim is to handle errors of any origin (physical faults, development faults, operator faults)

What is Software Fault Tolerance?

The term ”software fault tolerance” can mean two things:

1. ”the tolerance of software development faults”, or
2. ”the tolerance of faults by the use of software”

Definition 1 is more commonly used.
Definition 2 is used by N. Storey (author of the course book).

The term ”software redundancy” corresponds to definition 2.

Design Diversity

Design diversity is used to tolerate development faults in hardware and software

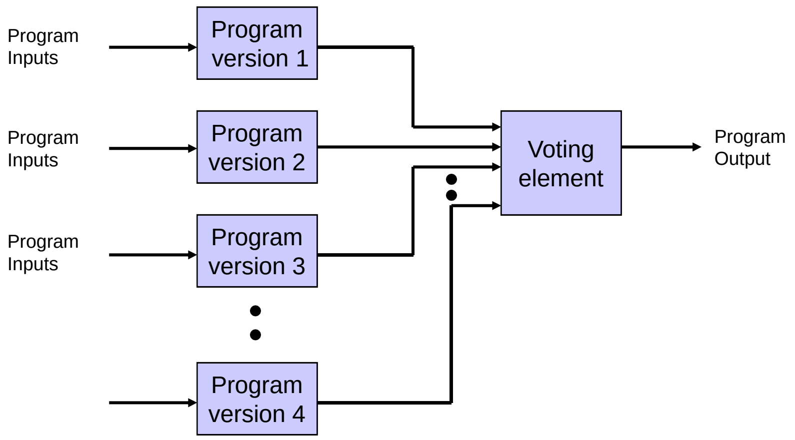
Two techniques for tolerating software design faults:

- N-version programming
- Recovery blocks

N-version programming

- Uses majority voting on results produced by N program versions
- Program versions are developed by different teams of programmers
- Assumes that programs fail independently
- Resembles hardware voting redundancy

N-version programming



Ensuring independence in N-version programming

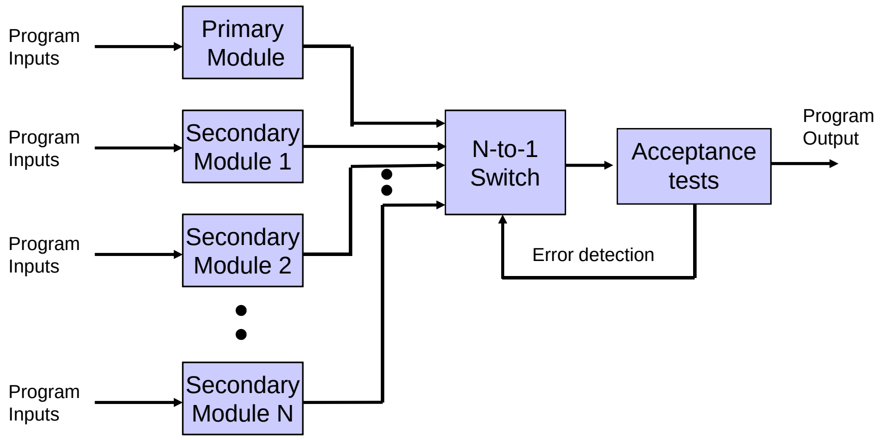
- Use different design teams for each version
- Use diverse specifications
- Prevent cooperation among design teams
- Use diverse programming languages, compilers, CASE tools, etc.
- …

Recovery Blocks

- Uses one primary software module and one or several secondary (back-up) software modules
- Assumes that program failures can be detected by acceptance tests
- Executes only the primary module under error-free conditions
- Resembles dynamic hardware redundancy

Lecture 7 EDA122/DIT061 Fault-Tolerant Computer Systems / DAT270 Dependable Computer Systems 37

Recovery blocks



Lecture 7 EDA122/DIT061 Fault-Tolerant Computer Systems / DAT270 Dependable Computer Systems 38

Fault tolerance in the Airbus A330/A340 fly-by-wire system

- Motivation
- System overview
- Design diversity

Motivation for fly-by-wire system

- Improving safety through automated control
  - Reducing the pilot's workload
    - 60% of air traffic accidents are due human errors of some kind (not only pilots errors).
    - Reduced workload for the pilot increases safety
  - Prevent the pilot from inadvertently exceeding the aircraft's controllability

Flight control surfaces of A340

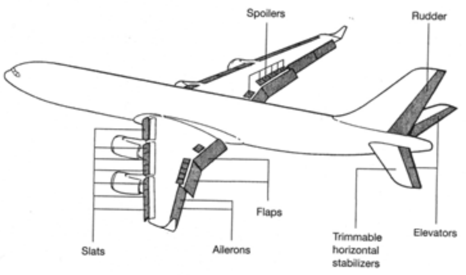


Figure 15.10 The flight control surfaces of an A340.

- Primary flight control surfaces
  - Airelons - controls the roll axis
  - Elevators – controls the pitch axis
  - Ruder – controls the yaw axis
- Secondary flight control surfaces
  - Flaps – lowering the flaps increases drag (air resistance) and lift
  - Spoilers increases drag and reduces lift
  - Slats – prevents stalls

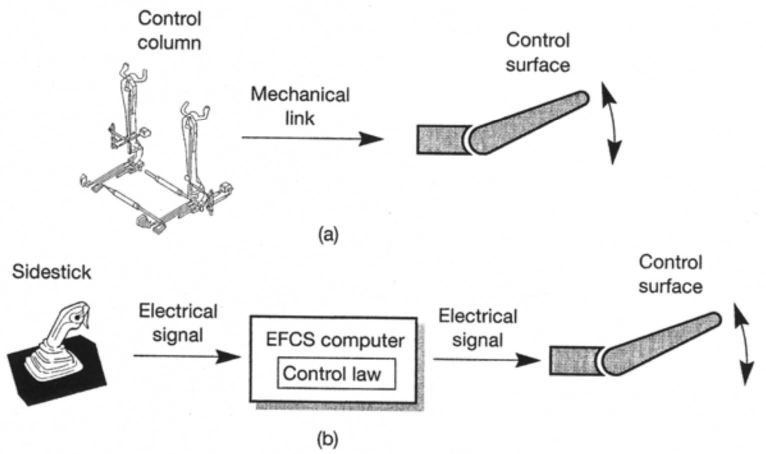
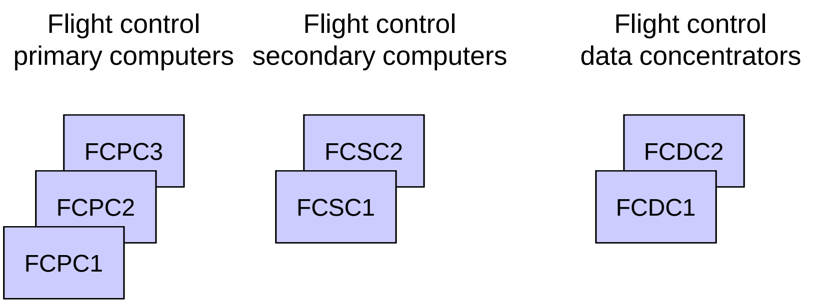


Figure 15.12 A comparison of (a) mechanical and (b) electrical control.

Design Diversity in Airbus A330/A340

- Two types of computers
  - 3 primary computers
  - 2 secondary computers
- Each computer are internally duplicated and consists of two channels
  - Command channel
  - Monitor channel

Architecture for A330/A340



Five computers control the flight control surfaces

Two computers control displays and warnings

Design Diversity in Airbus A330/A340

- Implementation of primary computers
  - Supplier: Aérospatiale (HW&SW)
  - Hardware: Two Intel 80386 (one for each channel)
  - Software: assembler for command channel, PL/M for monitor channel.
- Implementation of secondary computers
  - Supplier: Sextant Avionique (HW), Aérospatiale(SW)
  - Hardware: Two Intel 80186 (one for each channel)
  - Software: assembler for command channel, Pascal for monitor channel.

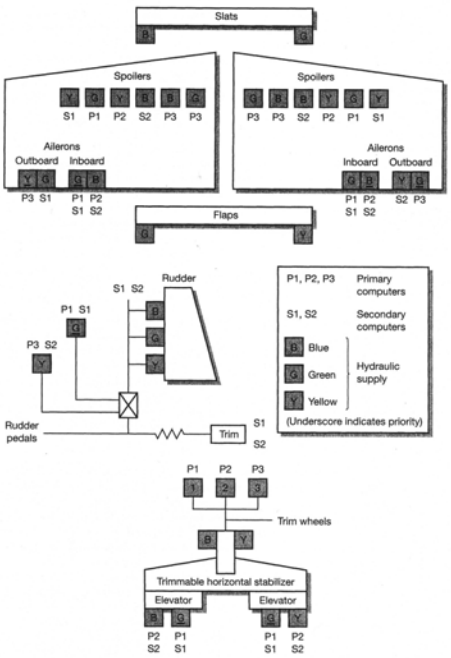


Figure 15.14 The computer and actuator arrangement of the A330/A340.

Principle of Graceful Degradation

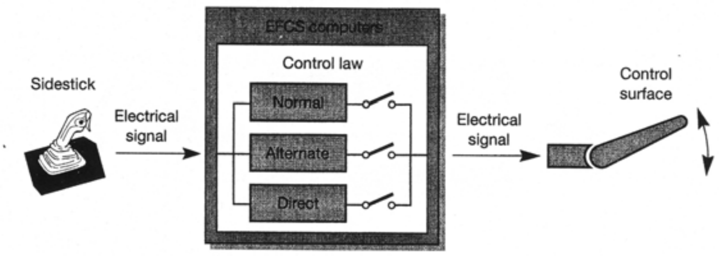


Figure 15.15 The flight control laws.

Normal vs. direct control

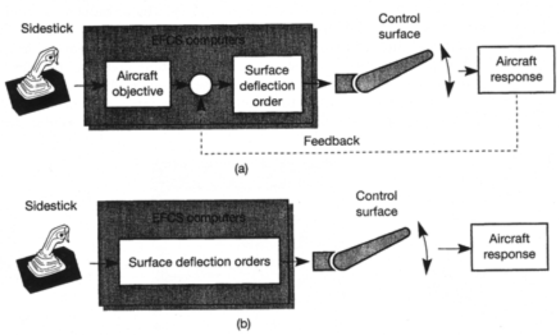


Figure 15.16 A comparison of the normal and direct control laws: (a) normal control law with aircraft feedback; (b) direct control law.

Summary of fault tolerance features in A330/A340

- Mechanical back-up: Mechanical linkages to the rudder and trimmable horizontal stabilizer give control in the event of total electronic system failure
- Computers: Five computers of two types with diverse hardware and software
- Sensors: Dual or triple redundant sensors
- Actuators: Single, double or triple actuators
- Hydraulic supplies: Three independent circuits and five pumps; hydraulic power can be produced by engines and ram air turbine
- Electrical supplies The A340 uses six generators and two batteries; four generators are driven by the engines, one by a auxiliary power unit (APU) and one by the hydraulic system.

Lecture 7 EDA122/DIT061 Fault-Tolerant Computer Systems / DAT270 Dependable Computer Systems 49

Ram air turbines

Lecture 7 EDA122/DIT061 Fault-Tolerant Computer Systems / DAT270 Dependable Computer Systems 50

Overview of Lecture 8

- Fault tolerance in space computers
  Guest lecture by Torbjörn Hult, RUAG Aerospace Sweden (formerly Saab Space)

Preparations:

- Ariane 501 failure report
- The US space shuttle's computer system, page 152 -154 in the course book
- Lecture slides

Overview of Lecture 9

- Management
- Life-cycles models
- Standards
- Safety case
- Verification and Validation
- Fault-tree analysis
- Failure mode effects analysis

Preparations:

- Lecture notes
- Chapter 3 - 5 in the course book, see reading instructions on home page.