THE CHALLENGE OF DEPENDABLE SYSTEMS – CSC 8201

ASSESSMENT 2022-23

RELIABILITY AND FAULT-TOLERANCE

1. In the given question car is the system, the driver will be the environment for the system, and the gear component will be the system's behaviour which interacts with the system using the environment.

The term “Erroneous state” refers to the broken-down state of the reverse gear.

Errors are defined by the system's internal state as undesirable being a Bad state.

Multiple lower-level errors can be called a fault by the system.

The car is the system and the reverse gear is the component of the system it is never

used before, due to some undesirable behaviour happening within the system

causing the failure of the component in the system., It is because of the

manifestation of a defect within the system or some Bad event that happened inside

the system that causes the failure., The internal state of the system is

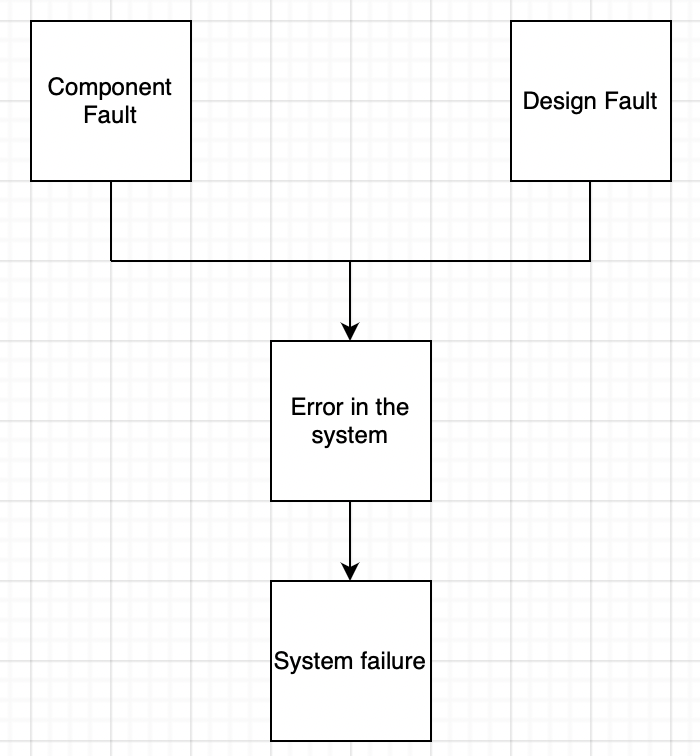
undesirable (Bad state) called an error, it might be the behaviour of the system is

incorrect as per the specification then the failure happens then we need to seek an

internal investigation for the cause of the failure.

Failures can occur due to errors inside the system, failure is an undesirable behaviour by the system, which we can observe from the outside of the system, i.e., by the driver, in this case, the driver can observe the failure from outside that the reverse gear of the car is in a broken-down state.

Generally, system failure occurs either by design faults or by component faults, these faults can be called an error in the system.



1. In the given figure there are four components C1, C2, C3, and C4 and one voter V located in the system, Voter V is a majority voter it takes the majority value of the input and produces the output

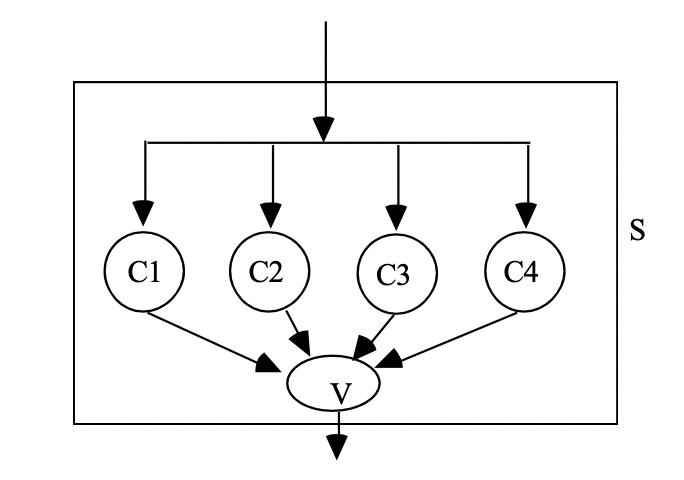


Figure 1

Consider all the components C1, C2, C3, and C4 produce the same result until the 90th input, after C3 becomes faulty during the 90th input and it produces the value of 0 to the voter V, voter then compares the value coming from C3 with all other components which produce the output value of 1 and then it takes the majority output from the components and produces the result as 1 and considers the C3 as a fault component as shown in figure 2.

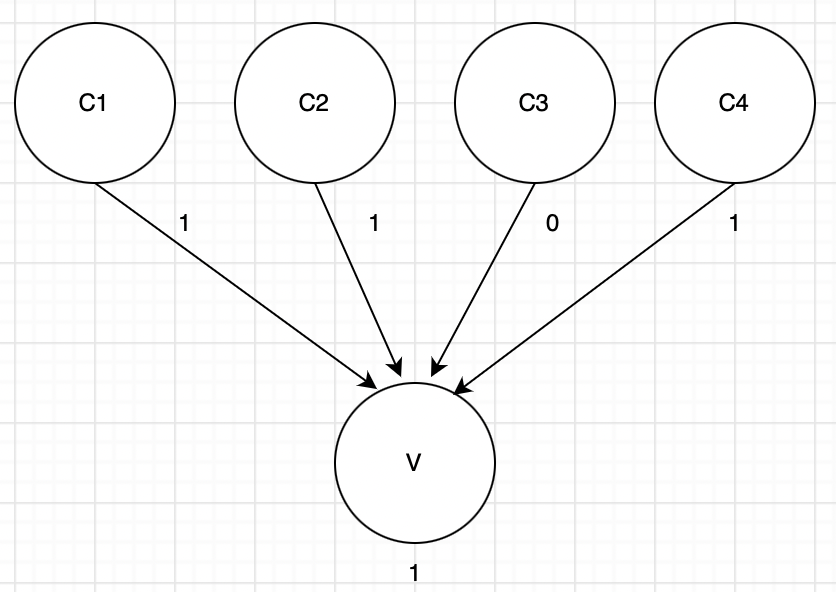


Figure 2

Voter V acts as a majority voter and it compares the value coming from all other components in the system, if component C3 produces identically incorrect results after the 90th input then the voter will check the other components C1, C2 and C4

results, it considers C3 as a fault component and it creates a log to store the value of that component for future reference.

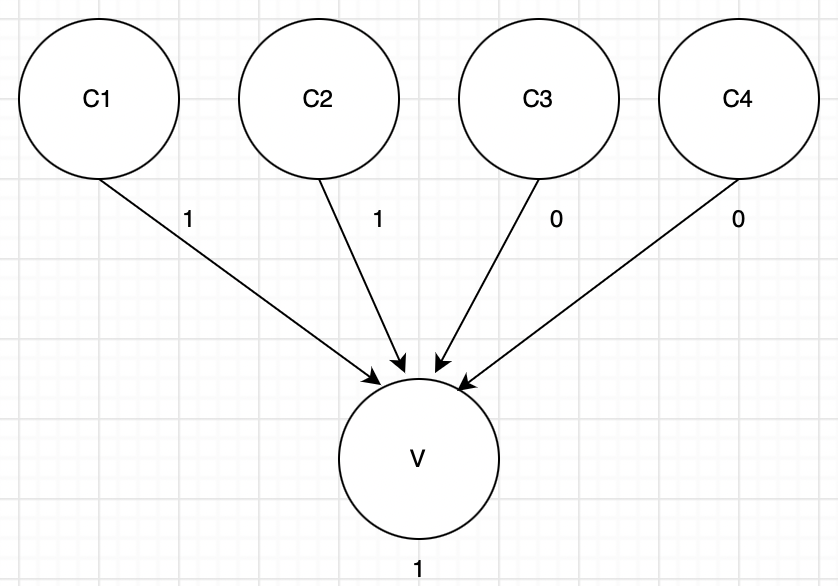


Figure 3

As per the given assumptions, if the component C4 fails at the 100th input, C3 already got failed in the 90th input and produces the output 0, then the voter identifies the faulty component using the log data, it produces the output as 1 and considers both C3 and C4 are the fault components.

If suppose the components C2, C3 and C4 fail and produces the output as 0, then the voter waits until the result from C1, after it produces the output as 1 it compares the value with all other components and takes the majority value and produces the output as 0 as shown in figure 4.

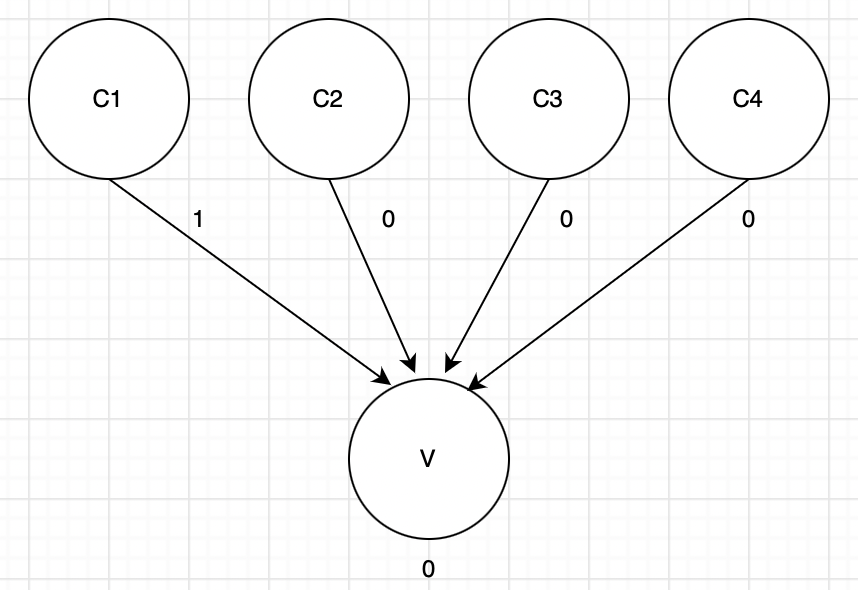


Figure 4

1. Given a=1; b=2; c=0;

ensure (acceptance test for recovery block) ---- Establishing an incremental checkpoint

In the given question we have an **ensure** (acceptance test for the recovery block), **by** (primary module) and **else by** (secondary module)

While executing the program if **by** (primary module) fails then restore the **ensure** (recovery block) and then run the system again for the **else by** (secondary module) if none of the modules produces the output then reset the **ensure** (recovery block) and produce the else error.

Recovery block syntax

ensure - acceptance test

by - primary module

else by - secondary module

else error – error

Given

a = 1; b = 2; c = 0;

[ R1 a=c+4; α

Record:

a was 1;

[ R1 a=c+4; b=6; β

Record:

a was 1;

b was 2;

[R1 c=5; γ

Record:

c was 0;

1. A recovery line is also called a consistent cut.

Given L1 = {R11, R21, R31, R41} is a recovery line.

Given {R11, R12, R21, R22, R31, R32, R41, R42} is an active recovery point.

P1, P2, P3, and P4 is the process of the system.

If the cut is consistent then the information should flow as per the below 2 cases

CASE 1: L.H.S w.r.t the consistent cut in P1.

R.H.S w.r.t the consistent cut in P2.

CASE 2: R.H.S w.r.t the consistent cut in P1.

R.H.S w.r.t the consistent cut in P2.

The consistent cut should pass through any one of the active recovery points in each processor.

As per the definition given if the cut is to be consistent then the information should

flow from the before-set (past) of process P1 to the after-set (future) of process P2

w.r.t the recovery line of P1 and P2, also it can flow from the after-set (future) of

process P1 to the after-set (future) of process P2 w.r.t the recovery line of P1 and P2,

but no information should pass from the after-set (future) of P1 to the before-set

(past) of process P2 and before-set (past) of process P1 to the before-set (past) of

process P2 w.r.t the recovery line of P1 and P2.

There can be more than one consistent cut that can pass through one recovery point.

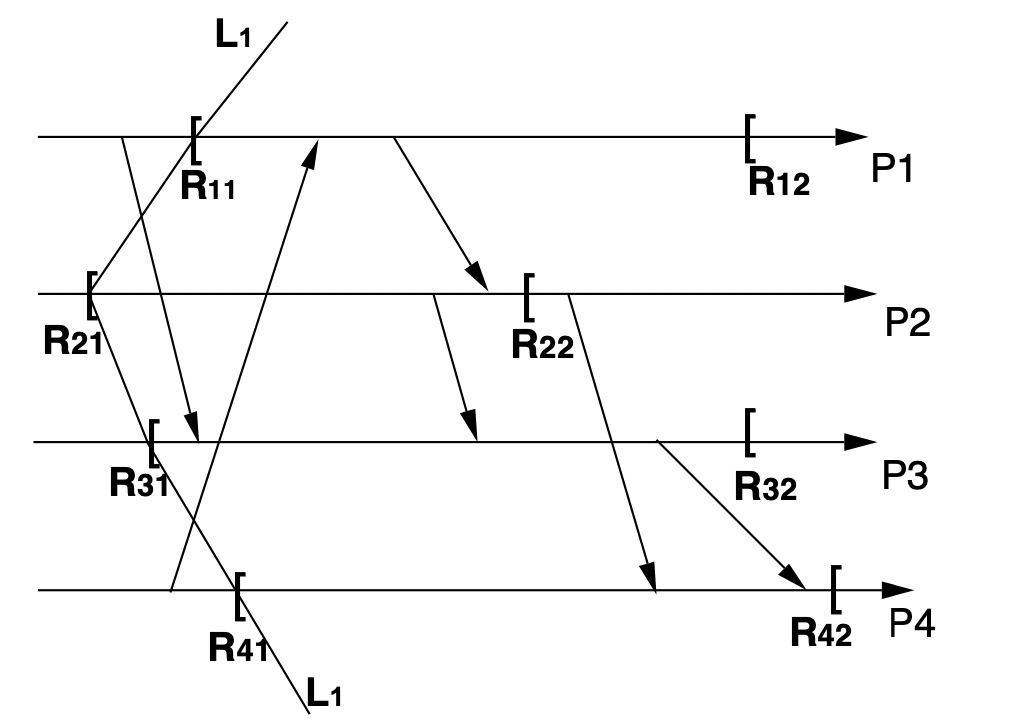


Figure 5

1. From the above figure, it shows that there are two information flows from P1 to P3 and P4 to P1, information 1 flows from the before-set (past) of process P1 to the after-set (future) of process P3 w.r.t the {R11, R31} recovery points and information 2 flows from the before-set (past) of process P4 to the after-set (future) of process P1 of the system that concludes given L1 = {R11, R21, R31, R41} is a recovery line.
2. As per the above figure, there is no recovery line passing through R32.
3. The reason why {R12, R22, R42} cannot be a recovery line is that,

Consider {R12, R22}, the above figure clearly shows that only one piece of information flows from the before-set (past) of process P1 to the before-set (past) of process P2 w.r.t the {R12, R22}, So to pass a recovery line between R12 and R22 the information should pass either from past to future or from future to future w.r.t the {R12, R22}, so we cannot draw a recovery line between R12 and R22.

On the other side, consider {R22, R42}, from figure 5, to draw the recovery line for {R22, R42} it should pass through R32, as mentioned earlier to draw a recovery line it should pass through any one of the active recovery points from each process.

1. In the diverse design of Air Bus systems, the design strategy of graceful degradation works with the two computational channels each will be having one CPU and input/output devices, Only one of the two channels will produce the output other will monitor the one which produces the output.

Each computer has been programmed in such a manner that it can shut down the whole computer if it causes some error, they have designed the system with two different channels having different programs installed.

The system is designed in such a way that if any one of the programs fails it will get detected and compare the output of the two programs and if the output produces a value above the threshold then the system shuts itself down and the secondary processor will take the control of the system.

In this Airbus system we have two processors “command” and “monitor” which was having two different programs it will always compares the output of each processor and detect the error from the output it produces, not only comparing the outputs of the program it also uses watchdog techniques, the execution regularly needs to go the watchdog and reset the timing, if the timer is not reset some of the processes may be struck in the infinite loop, it also got acceptance and exception testing, In acceptance testing the error detection happens and in exception testing system will handle the unwanted event occurs when the system runs the program.

The graceful degradation strategy is to have some limited functionality work if there is any failure happens in the system or multiple errors occur, Airbus design has two roll controls A310 Roll Control and A320 Pitch Control both are designed in such a way that if one fails another component will take the control of the system.

In the given A310 Roll Control there are four components involved each of them having two computational channels, aircraft is controlled by the pair of roll axis ailerons and spoilers. It can be controlled either by manual or automatic mode, Automatic mode is controlled by two FCC “Flight Control Computers” (FCC1 & FCC2), Only one can function at a time, the other is being used as a spare, if both fail then the pilot will take control manually, Therefore the automatic flight control is not dangerous (except during a short period of automatic landing during bad weather conditions)

Spoilers are used for the manual control mode, and two “Electrical Flight Control Units” (EFCU1 & EFCU2) are used to control the spoilers, if both of them got failed still the pilot can control the system using ailerons as spoilers are no more available to use.

1. In A320 Pitch Control there are four computers are used to control the aircraft's pitch axis ELAC1 (Elevator and Aileron Computer), ELAC2, SEC1 (Spoiler and Elevator Computer), and SEC2.

Timing check happens by other computers, if one computer will be having full authority and is in charge used to send the message “I am alive” to other computers if the computer fails it shuts itself down, other computers use to monitor whether the system running out of time or sending the message “I am alive” on time. It also detects the failure and based on the priority, one will take the control of the pitch axis.

1. The “command” and “monitor” are often of the same type. Because of the two different programs, It is very likely that a microprocessor design error would behave as a software error of one of the programs. More, as the used processors are hardened versions of commercial ones, it can be argued that they are extensively tested.

The tolerance to design faults is achieved using electro-mechanical back-up, or a design fault-tolerant computing system, as for the “fly-by-wire” A320.