Dependable Software System

Sociotechnical Systems :-

- System that involve people, technology and organizational structure
- Why important?

Critical systems depend on both humans and technology

- Healthcare
- Air traffic control
- Banking

Key components of sociotechnical systems :-

- People → Users, operators, stakeholders
- Technology → Hardware, software, infrastructure
- Processes → Workflows, policies, procedures
- Organization → Culture, management, communication structures
- Environment → External influences like regulations, market conditions.

Human Error:-

Two approach : ① Person Approach : (fault of individual)

(Assumes, errors can be eliminated by controlling behaviour)

(Soulution: Discipline, retraining, strict procedures)

② Systems Approach :

Errors are inevitable. Errors result from system design and organizational factor.

Solutions: Barriers, safeguards, recovery mechanisms.

Important of Systems Approach:

Improve security and dependability by
- adding barriers and defenses
- Designing processes that mitigate human error
- Implementing automated checks

Examples: Automated Conflict Alert (Detect potential collisions and sound alarm)
Banking Fraud Detection flags suspicious transaction

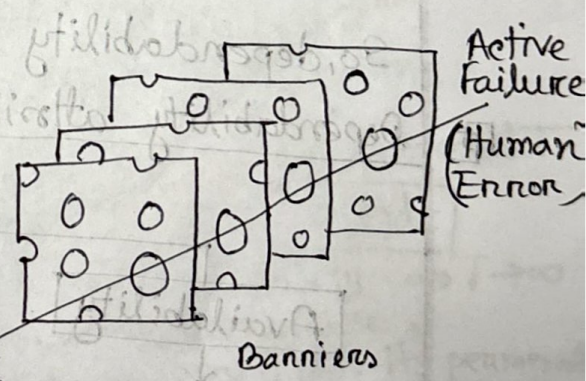
Weakness of Defences

These weaknesses are called latent conditions. Because they usually only contribute to system failure when some other problem occurs.

Examples:
- Conflict alert system may produce many false alarms
- Controllers may therefore ignore warnings from the system

Swiss Cheese Model of System Failure:

Defences = slices of Swiss cheese
Latent condition = holes in barriers
Failures occur when holes align, allowing errors to pass through



Reducing Probability of failure:

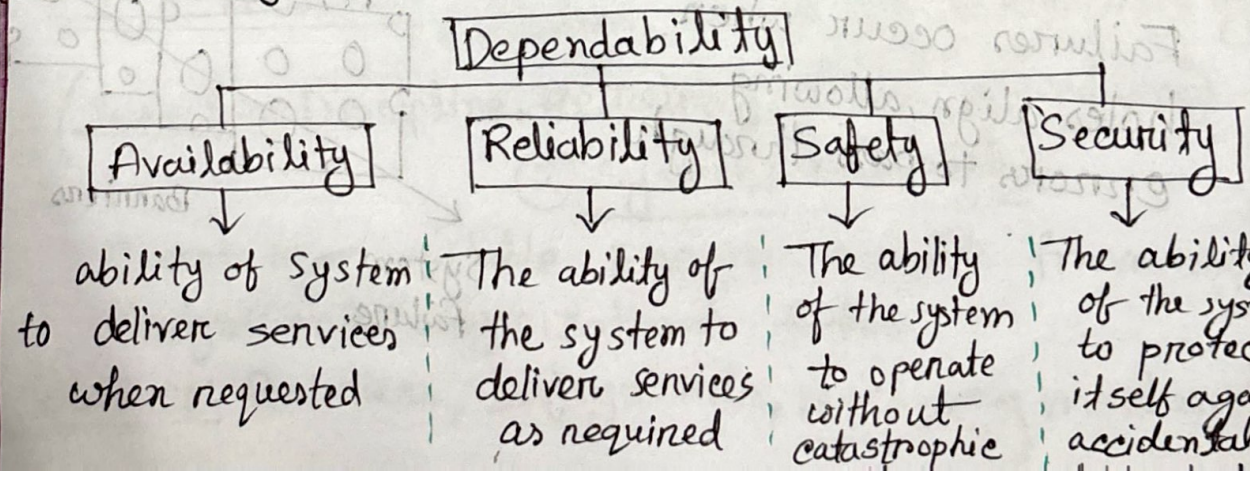
- Include different types of barriers
(Holes will be in different places, so less chance of error)
- Minimize latent conditions
(Reduce number of weaknesses in the system)
- Optimize system and process design
(Prevent active failures by reducing workload, information overload and stress).

Software Dependability

- Dependability is the ability of a system to deliver its intended level of service to users.
- Its important for —
  - Safety-Critical Applications (aviation, healthcare)
  - Mission-Critical " (space, military)
  - Business-Critical " (banking, e-commerce)

So, dependability

Dependability attributes / Properties:-



Reliability:

Reliability $R(t)$ of a system at time t is the probability that the system operates without a failure ~~int~~ in the interval $[0,t]$, given that the system was performing correctly at time 0.

- Measures continuous delivery of correct service.
- Important for
  - Life-critical System (heart pacemaker) → Must function without failure
  - Remote/Unreachable System (deep-space probes) → Maintenance is impossible
- Reliability is a function of time
  - Hardware Systems: (measured in ~~natural~~ calender time, hours)
  - Software system: " in natural units (transactions, queries, jobs)

Availability:

It is the probability that the system is operational ~~and accessible when~~ / functioning correctly at the instant of time t.

Types of availability

1. Point availability (Instanteous) : $A(t)$ at a specific time → exact time
2. Mission " : Avg availability over an interval T → $A(T) = \frac{1}{T}\int_0^T A(t)dt$
3. Steady-state " : Long term availability as $T \to \infty$

Relation to Reliability:

- If system cannot be repairable, $A(t) = R(t)$, Because once it fails its permenantly unavailable.
- For non-repairable systems, steady-state availability → 0 as $T \to \infty$

Eventually system fails down.

• Steady state availability $A(\infty)$ is often specified in terms of downtime per year

example: 99% availability ⟶ 3.65 days/year downtime

• Availability is typically measure of dependability for systems where short interruptions can be tolerated.

examples —

Networked systems: Telephone switching, web server

Power Systems: Load-Shedding

Safety: Safety $S(t)$ is the probability that the system will:

— Perform its intended function correctly

— OR, Shut down or fail in a safe manner (fail-safe)

During time interval $[0, t]$, given that the system was operating correctly at time 0.

• Reliability treats all failures equally.

• Safety distinguishes in —

① Fail-Safe: A failure that does not cause harm

example: Alarm false positive (when no danger)

② Fail-Unsafe: A failure that leads to hazardous consequences.

example: Alarm false negative (during danger)

- Why safety matters?

Crucial in safety-crucial systems, where failure can lead to human injury, loss of life or environmental disaster.

- Examples - Trains, Automobiles, Avionics, Medical devices, Military systems.

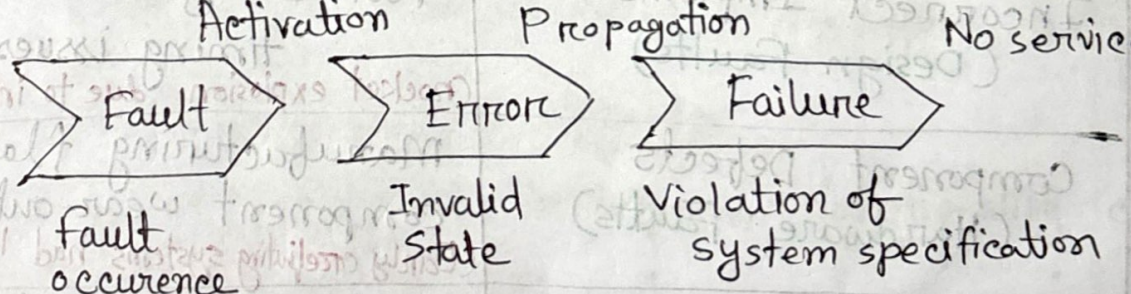
Security: Security refers to the ability of a system to protect itself from unauthorized access, tempering or disruption.

Confidentiality: Ensures that sensitive data is not exposed to unauthorized users.

Integrity: Protects data from unauthorized modification or corruption.

Availability: Ensures that the system remains accessible and functional when needed.

Dependability Threats/Impairements:



• **Faults:** Defects or flaws in a hardware or software component

example → The altimeter sensor of an aeroplane malfunctions, providing inaccurate altitude readings

• **Errors:** Deviation from accuracy in computation, which occurs as a result of a fault.

example → The autopilot system receives incorrect data, causing it to incorrectly adjust altitude

• **Failures:** Non-performance of some action which is due ~~to~~ or expected.

example → Altitude deviation leads to turbulance, uncomfortable passenger experiences or triggers safety mechanisms for manual control.

☐ Four majour Sources of Faults

Incorrect Specification	Faulty algorithms, missing requirements (Failed due to unit mismatch in specification)
Incorrect Implementation (Design Faults)	Poor coding, incorrect logic, timing issues (rocket explosion due to integer overflow)
Component Defects (Hardware Faults)	Manufacturing flaws, component wear-out (Early computing systems had low-reliability components.)
External Factors	Environmental (radiation, vibration) human actions. Radiation flipping memory bits, cyberattacks

Common-Mode Faults:

- A fault that occurs simultaneously in two or more redundant components due to shared dependencies
  - Design Diversity is the solution:
    → Implementation of more than one variant of the function to be performed
    → Better to vary a design at higher levels of abstraction

- Examples — "Design Diversity"
  - Use different algorithms
  - Use multiple programming languages
  - Separate design teams, rules and tools.

Software Faults:

Software is deterministic, behaves same way under the same condition. So, it does not wear out like hardware.

- Main sources of software faults:
  - Design Faults
    → Primarily caused by human factors (inconnect logic, poor requirement)
    → Harder to prevent compared to hardware faults.
  - Faults introduced by Upgrades
    → Upgrades intended to improve functionality can have unintended consequences.

Examples — 3 lines code change in multi-million line program caused telephone outage.

Dependability Means:

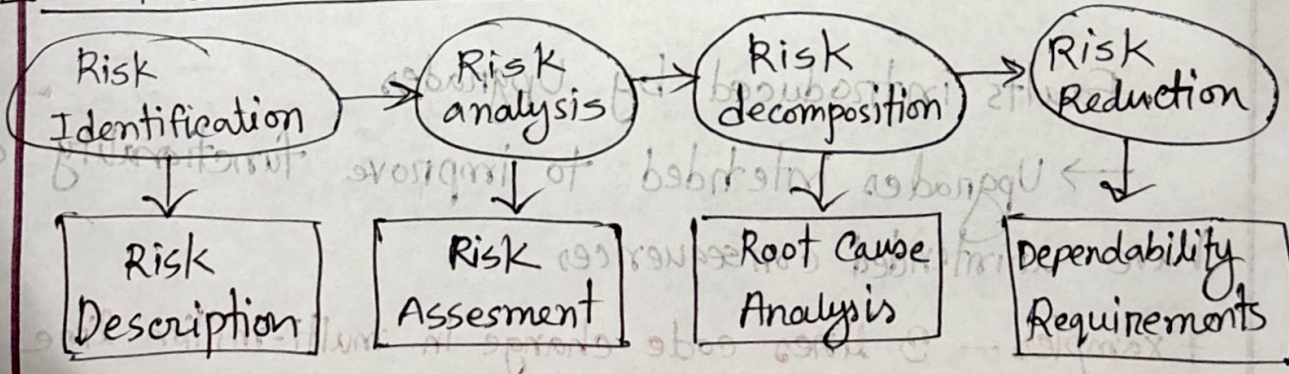
- Fault Tolerance — Ensures system functionality despite faults.
- " Prevention — Prevents occurence of faults
- " Removal — Reduces existing faults
- " Forecasting — Estimate number of faults and their impacts

Fault Tolerance: (continued)

- Redundancy: Extra components to ensure ↑ function
- Fault Masking: Hides faults from the system output (error-connecting code)
- " Detection: Identifies faults via comparison of redundant components
- " Location: Identifies where the fault occured
- " Containment: Isolates faults to prevent spread.
- " Recovery: Grateful degradation or use of "backup" components

Dependability Requirement Specification

Risk-driven Requirement specification:



Risk Analysis Phases:

- Preliminary Risk Analysis — identify external risks (environmental factors)
- Life-cycle " " — Address design-related risks
- Operational " " — Handle user interface (UI) and operator errors.

Safety Requirement Specification:

- In Safety-Critical Systems, Failures may cause injury or death
- Focus: Minimize failure probability
- Safety Vs Functionality: Balance protection without over-restricting operation
- Key Concepts:
  - Hazard — A potential source of harm
  - Risk — Probability of system entering a hazardous state

Risk-based Safety Requirements Specification:

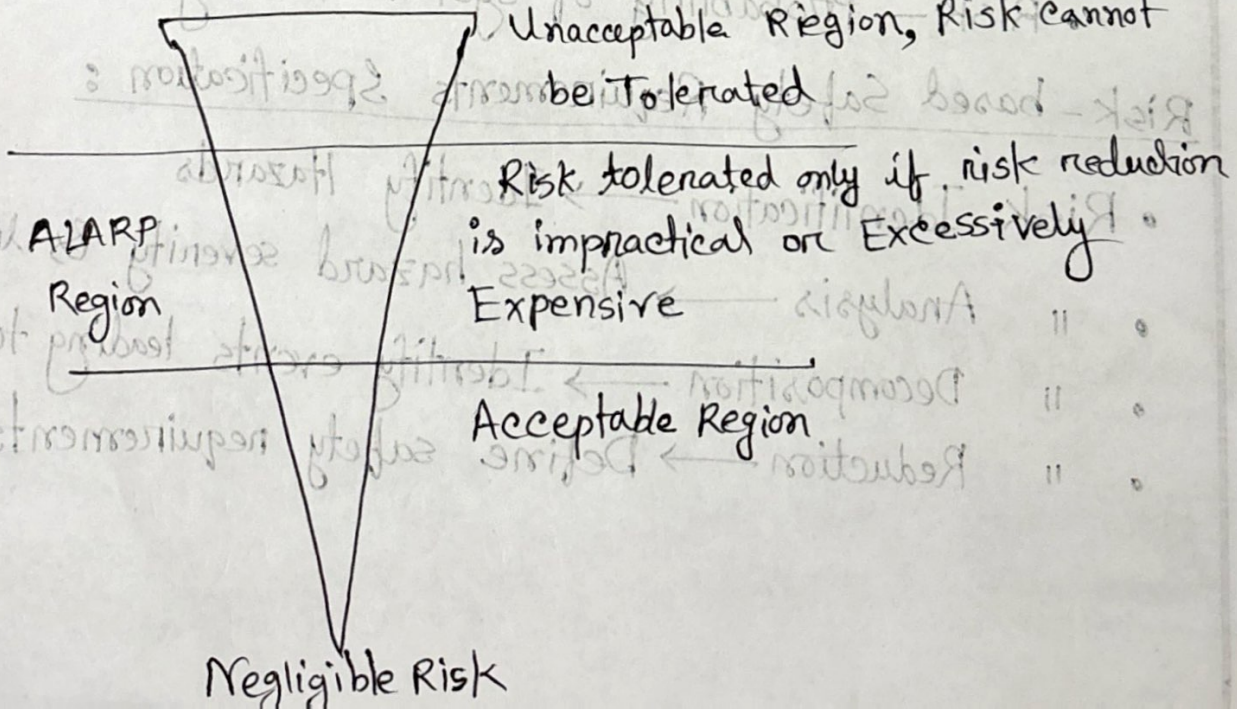
- Risk Identification ⟶ Identify Hazards
- " Analysis ⟶ Assess hazard severity & likelihood
- " Decomposition ⟶ Identify events leading to hazards
- " Reduction ⟶ Define safety requirements.

Hazard Identification:

- Identify different hazard types
  → (physical, electrical, biological, service failures etc)
- Example: Insulin pump system hazards:—
  → Insulin overdose/underdose (service failure)
  → Power failure (electrical)
  → Incorrect fitting (physical)

Hazard Assessment:

- Categorizing risks:
  → Intolerable — Must be eliminated (insulin overdose)
  → ALARP —
  → Acceptable — Minor impact
- Risk Triangle: Cost of Risk Reduction Vs Risk severity



Identified Hazard	Hazard probability	Accident severity	Estimated Risk	Acceptability
1. Insulin over dose computation	Medium	High	High	Intolerable
2. Insulin underdose	"	Low	Low	Acceptable
3. Failure of Hardware monitoring system	"	Medium	"	ALARP
4. Power failure	High	Low	"	Acceptable
5. Machine incorrectly fixed	"	High	High	Intorelable
6. Machine breaks in patient	Low	"	Medium	ALARP
7. Machine causes infection	Medium	Medium	"	"
8. Electrical interference	Low	High	"	"
9. Allergic reaction	"	Low	Low	Acceptable

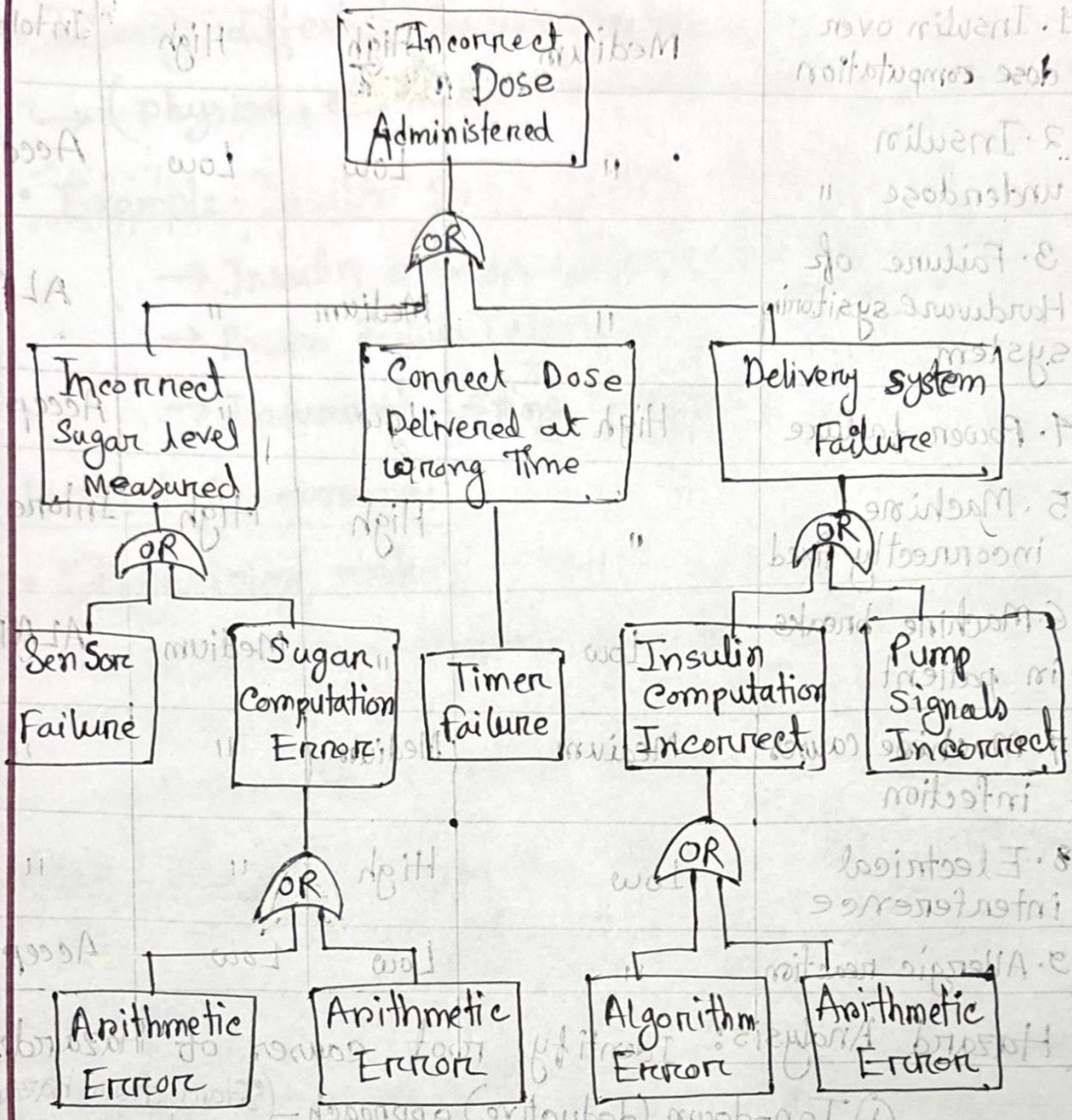
Hazard Analysis: Identify root causes of Hazards using -

① Top-down (deductive) approach - (Start from hazard and trace causes)

② Bottom-up (inductive) approach - (Start from failures and identify hazards)

• Fault Tree Analysis (FTA) is a top-down (deductive) approach.

Fault Trees:-



Risk Reduction Strategies:-

- Hazard Avoidance – Design system to prevent hazard from occuring.
- (Hazard Detection & Removal) – Detect and neutralize hazards before they cause harm
- Damage Limitation – Minimize accident consequences.

Designers of critical systems use combination of these approaches (Hazard Avoidance, Hazard Detection & Removal, Damage Limitation)

- Example - Chemical Plant Safety System
  - → Detect and avoid high pressure
  - → Independent protection system (relief valve) as backup

Example of Safety Requirements:

- SR1: The system shall not deliver a single dose of insulin that is greater than a specified maximum dose for a system user.
- SR2: The system shall not deliver a daily cumulative dose of insulin that is greater than specified maximum daily dose for a system user.
- SR3: The system shall include a hardware diagnostic facility that shall be executed at least four times per hour.
- SR4: The system shall include an exception handler for all the exceptions that are identified in Table 3.
- SR5: The audible alarm shall be sounded when any hardware or software anomaly is discovered and a diagnostic message, as defined in Table 4, shall be displayed.
- SR6: In the event of an alarm, insulin delivery shall be suspended until the user has reset the system and cleared the alarm.

Reliability Requirement Specification:

- System Reliability depends on hardware, software and operator reliability.
- Reliability differs from safety and security:
  → Measuring a desired level of reliability makes sense (failures per week etc).
  → Safety & security focus on preventing critical failures (even one failures is unacceptable)

Types of Reliability Requirements:

① Non-functional / Quantative: Specifies acceptable failure rates or system downtime.

② Functional: Defines mechanisms to detect, prevent and recover from faults.

Risk-Based Reliability Requirements Specification:

- Risk Identification → Identify failure types and potential ~~issues~~ losses
- Risk Analysis → Estimate costs & consequences of failures
- Risk Decomposition → Analyze root causes of critical failures
- Risk Reduction → Define quantitative reliability specifications & fault-handling mechanisms.

Reliability Metrics:

- **POFOD (Probability of failure on Demand)**: Likelihood of failure during a request.

→ Example: POFOD = 0.001 (1 failure per 1000 requests)

- **ROCOF (Rate of occurence of failures)**: Failures per unit time or per ~~inst~~ transaction.

→ Example: ROCOF = 2 failures per hour → MTTF (Mean Time To Failure)

$\frac{1}{ROCOF} = MTTF = 30$ min

- **AVAIL (Availability)**:

Probability that a system is operational when needed.

→ Example: 99.99% uptime = 8.4 sec downtime per day.

POFOD: (Probability of failure On Demand):

→ **Use case**: When failure on demand could lead to a serious system failure

→ **Example**: Protection Systems (e.g. a chemical reactor shutdown mechanism)

→ Why POFOD?

- Suitable for systems with infrequent demands
- Ensure critical failure prevention
- A low POFOD (e.g. 0.001) is acceptable if system demands are rare.

ROCOF: Rate of occurence of failures:

→ Use case: When the system is used regularly and failures need to be tracked over time.

→ Example: Transaction-based systems
(e.g. e-commerce platform, banking systems)

→ Why ROCOF?

- Measures failures over a specific period (e.g. failures per day)
- Helps maintain acceptable failure rates in high-usage systems.
- Can be defined per 1,000 transactions for precision.

MTTF: Mean Time To Failure:

→ Use case: When the absolute time between failure is critical

→ Example: systems with long running sessions
(e.g. CAD software)

→ Why MTTF?

- Ensures user don't lose progress due to unexpected failure
- The MTTF should be significantly longer than typical work sessions.

Availability (AVAIL):

→ Use Case: When short interruptions can be tolerated but overall uptime is critical

→ Examples: Power systems (load shedding), Networked System (web servers).

→ Why Availability?

- Measures the percentage of time of a system is operational
- Suitable for systems where occasional downtime is acceptable.
- Helps ensure minimal disruption of users.

Mean Time To Repair (MTTR):

- MTTR is the avg time required to repair the system.
- If system experiences n failures during its lifetime, then, total time that the system is operational is
  $= n \times MTTF$
- Similarly, the total time that the system is repaired is $= n \times MTTR$
- Relation to steady-state Availability –

$$A(\infty) = \frac{n\,MTTF}{n\,MTTF + n\,MTTR} = \frac{MTTF}{MTTF + MTTR}$$

*** Math (page-55)

Measuring Reliability:

- Failure logs – Track number of failures Vs service requests (POFOD)
- Time Between failures – Compute MTTF and ROCOF.
- Repair/Restart Time – Measure system recovery time (affects availability)

Time Measurement Units:

- Calender time : For continuous-operation systems
- Processor time : For event-driven systems
- Transaction count : For variable-load systems (e.g. banking)

Non-Functional Reliability Requirements:

- Quantitative specification of system reliability and availability
- Common in safety-critical systems, but increasingly used in business-critical systems.

Advantages of Quantitative Reliability Specification:

- Clarifies Stakeholders needs – Helps distinguish different failure types and associated costs
- Guides Testing Efforts – Defines when testing can stop based on reliability targets.
- Supports Design Decisions – Enables comparison of reliability-improving stategies.
- Facility Certification – Provides evidence for regulatory approval in critical systems

Overspecification Risks:-

- High development and validation costs
  → Testing system with POFOD = 0.0001 may require 50,000–60,000 test cases.
  → Availability figures (e.g. 0.999 vs 0.9999) may have minimal practical impact but huge cost differences.
- Difficult to translate stakeholder experience into metrics.
- Large-scale testing required to statistically validate reliability.
- High availability numbers may not reflect real world usage

Example: ATM networks prioritize availability over transaction reliability, as errors can be corrected later.

Avoiding Overspecification:-

- Categorize failures – Differentiate between minor and critical failures.
- Prioritize Key Services – High reliability of core services, lower for non-critical ones
- Use Alternative Dependability Mechanisms – Error detection, recovery methods instead of extreme reliability levels.

Case Studies of Non-Functional Specifications:-

(1) Banking ATM systems :-

- Critical component : Customer account database
  Availability → 0.9999 (downtime < 1 min/week)
- ATM Software
  → Lower availability (e.g 0.999) due to hardware and cash refill issues.
- Banks prefer availability over extreme reliability
  → Faulty transactions can be corrected later.

(2) Insulin Pump Reliability:-

- Transient Failures: fixable by users (e.g. recalibration)
  POFOD = 0.002 (1 in 500 demands, ~3.5 days).
- Permenant Failures: Require manufacturer intervention, POFOD ≤ 0.00002 (~1 per year)
- Safety vs Commercial Factors: Failures cause inconvenience, not immediate harm → focus on reducing service cost.

Functional Reliability Requirements:

Types of Functional Reliability Requirements:

- Checking Requirements: Detect invalid inputs before processing
- Recovery " : Define backup & restore mechanisms
- Redundancy " : Ensure single failures don't cause total system failure
- Process Requirements for Reliability:

→ Follow best practices to minimize faults in development
→ Leverage industry-specific knowledge for critical systems.

Example of Functional Reliability Requirements:

RR1: A pre-defined range for all operator inputs shall be defined and the system shall check that all operator inputs fall within pre-defined range (Checking).

RR2: Copies of the patient database shall be maintained on two separate servers that are not housed in the same building (Recovery, Redundancy)

RR3: N-version programming shall be used to implement the braking control system. (Redundancy)

RR4: The system must be implemented in a safe subset of Ada and checked using static analysis (Process)

Dependability: Modeling at the Design Phase:

- **Combinatorial Models**

→ Assume, that the failures of individual components are mutually independent

→ Include: Reliability Block Diagrams (RBDs), Fault Trees and Reliability Graphs.

- **Stochastic Models**

→ Consider the dependencies between components failures, enabling analysis of more complex scenarios.

MTTF, MTTR (Math)

given, MTTF = 2160h

MTTR = 36h

(a) Steady state Availability, $A(\infty) = \frac{MTTF}{MTTF+MTTR}$

$= \frac{2160}{2160+36}$

$= 0.9836$

(b) MTTR = 12h

$A(\infty) = \frac{MTTF}{MTTF+MTTR}$

$\Rightarrow 0.9836 = \frac{MTTF}{MTTF+12}$

$\Rightarrow 0.9836(MTTF + 12) = MTTF$

$\Rightarrow 0.9836MTTF + 11.8032 = MTTF$

$\Rightarrow MTTF - 0.9836\,MTTF = 11.8032$

$\Rightarrow 0.0164\,MTTF = 11.8032$

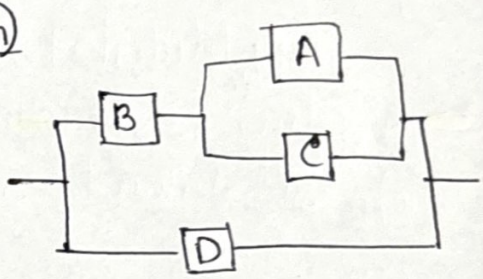
$\therefore MTTF = \frac{11.8032}{0.0164} = 719.707h$

RBD (Reliability Block Diagram)

(i) A and B work ; or

(ii) B and C work ; or

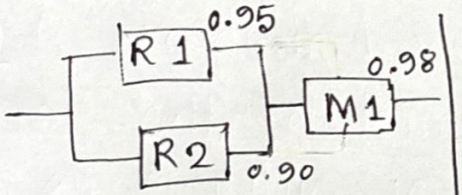
(iii) D work



ChatGPT (Ques Solve)

(Q1)

(a)



(b) $(0.95 \times 0.98) = 0.931$

Wrong answer

$1 - 0.931 = 0.069$

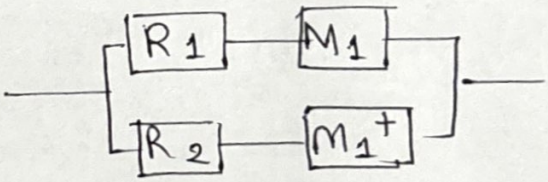
$1 - 0.90 = 0.1$

$(0.69 \times 0.1) = 0.069$

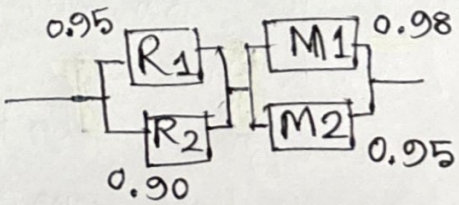
$1 - 0.069 = 0.931$

RBD = 0.931

(c) Here, the single point of failure is M_1



(d)



wrong answer

$0.95 \times 0.98 = 0.931 \rightarrow (1 - 0.931) = 0.06$

$0.90 \times 0.95 = 0.855 \rightarrow (1 - 855) = 0.145$

$1 - (0.069 \times 0.145) = 0.9899$