Architectural Foundations of Software Dependability

Reliability Engineering & The Mechanics of Failure

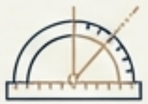
HIMMA CURRICULUM | BUILD YOUR PROOF

Key Question: Why does a system that meets every technical specification still fail in the eyes of the user?

The Landscape of Dependability

Dependability & Fault Management
The Abstract Concepts

Reliability Metrics
The Mathematics



Reliability Metrics
The Mathematics

Architecture & Patterns
The Build

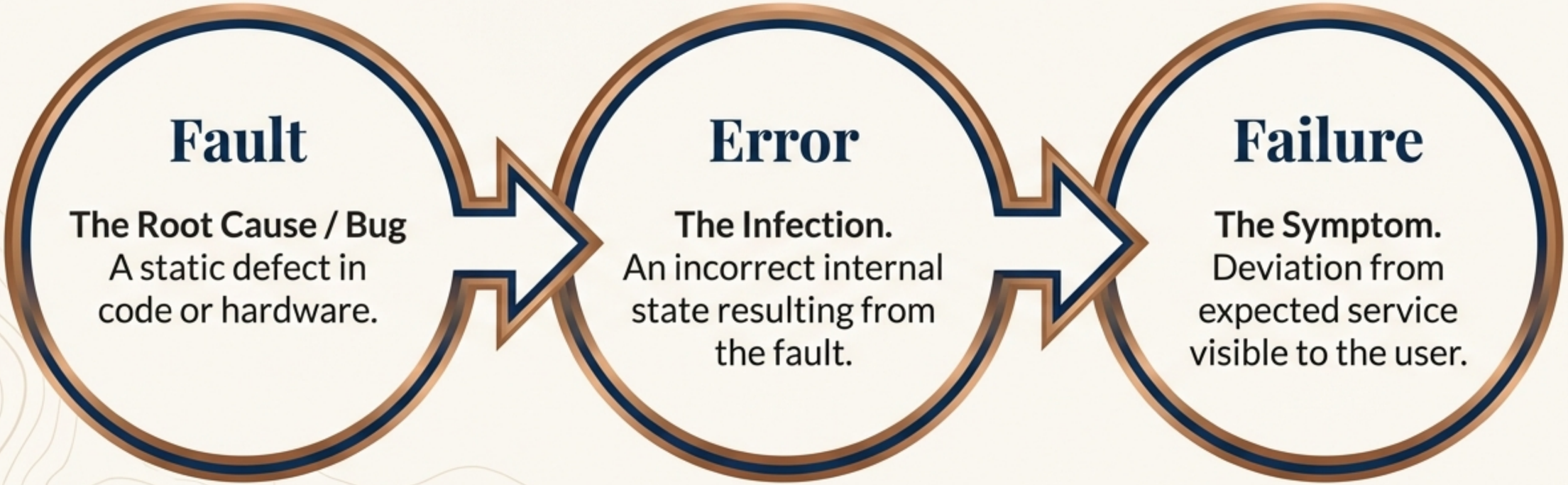
Measurement & Evidence
The Proof

Course Learning Outcomes (CLOs)

By the end of this lecture, you will be able to:

- [x] Distinguish between Reliability (Safety/Trust) and Availability (Uptime/Continuity).
- [x] Differentiate the failure chain mechanics: Fault → Error → Failure.
- [x] Select appropriate metrics (POFOD, ROCOF, AVAIL) based on system risk and usage patterns.
- [x] Propose architectural decisions (N-Version programming, Redundancy) to ensure dependability.
- [x] Apply the '8 Commandments' of reliable programming to code-level design.

The Anatomy of a Crash: The Chain of Failure



Critical Insight: Faults do not always cause errors; errors do not always cause failures. A system can recover before the user notices.

Reliability vs. Availability

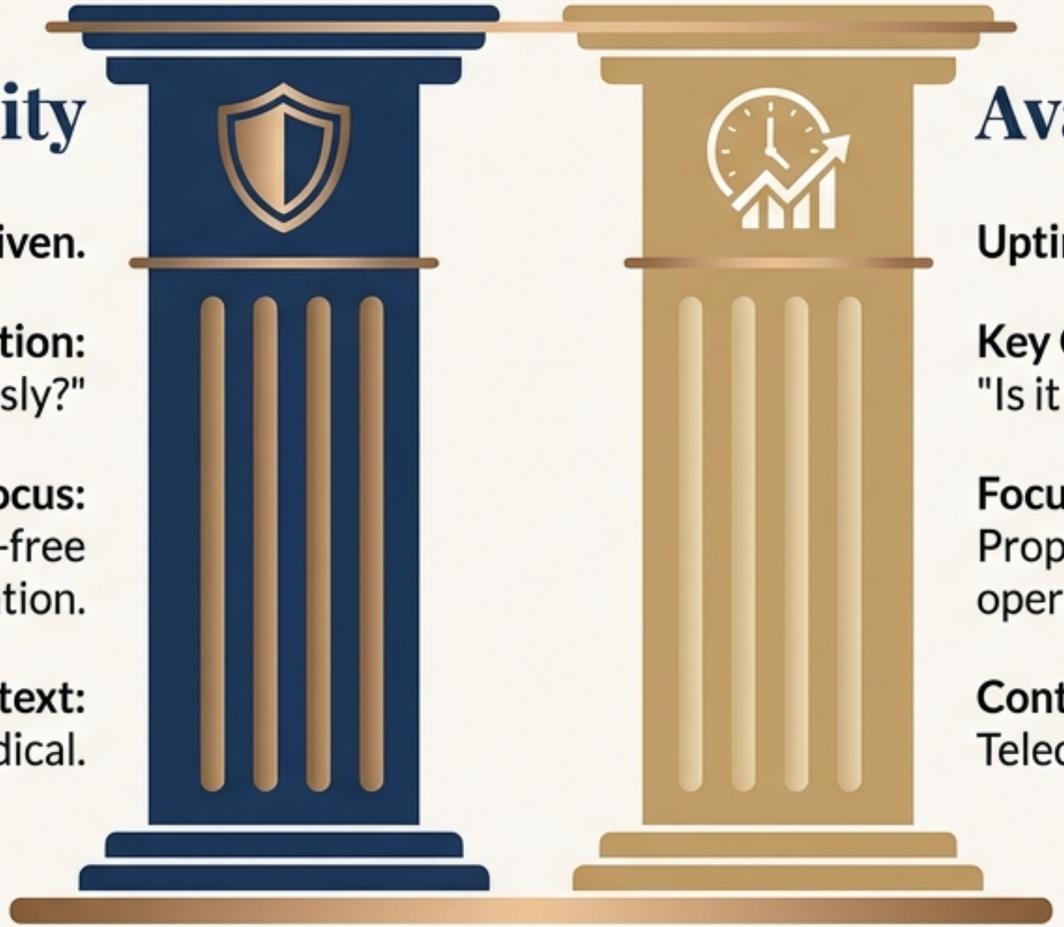
Reliability

Safety-Driven.

Key Question:
"Did it fail dangerously?"

Focus:
Probability of failure-free operation.

Context:
Aviation, Nuclear, Medical.



Availability

Uptime-Driven.

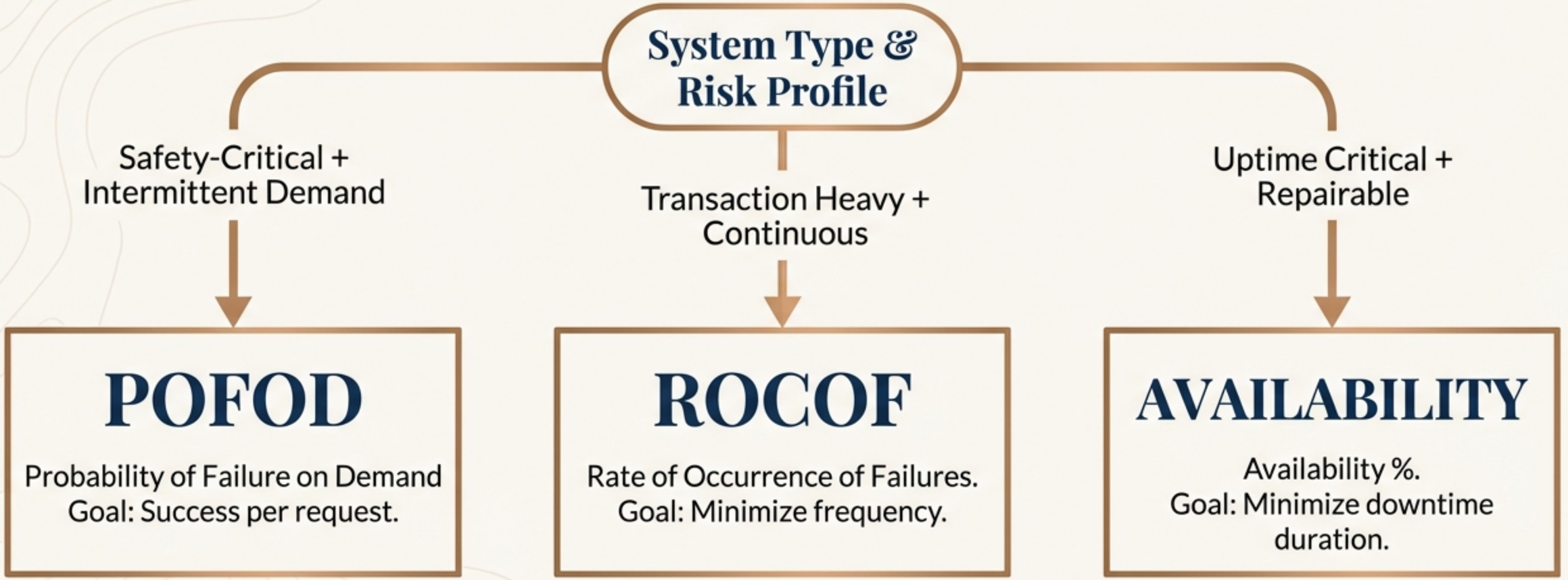
Key Question:
"Is it working right now?"

Focus:
Proportion of time operational.

Context:
Telecom, E-commerce, Banking.

Users judge Availability. Engineers must judge Reliability.

How to Measure the Unmeasurable



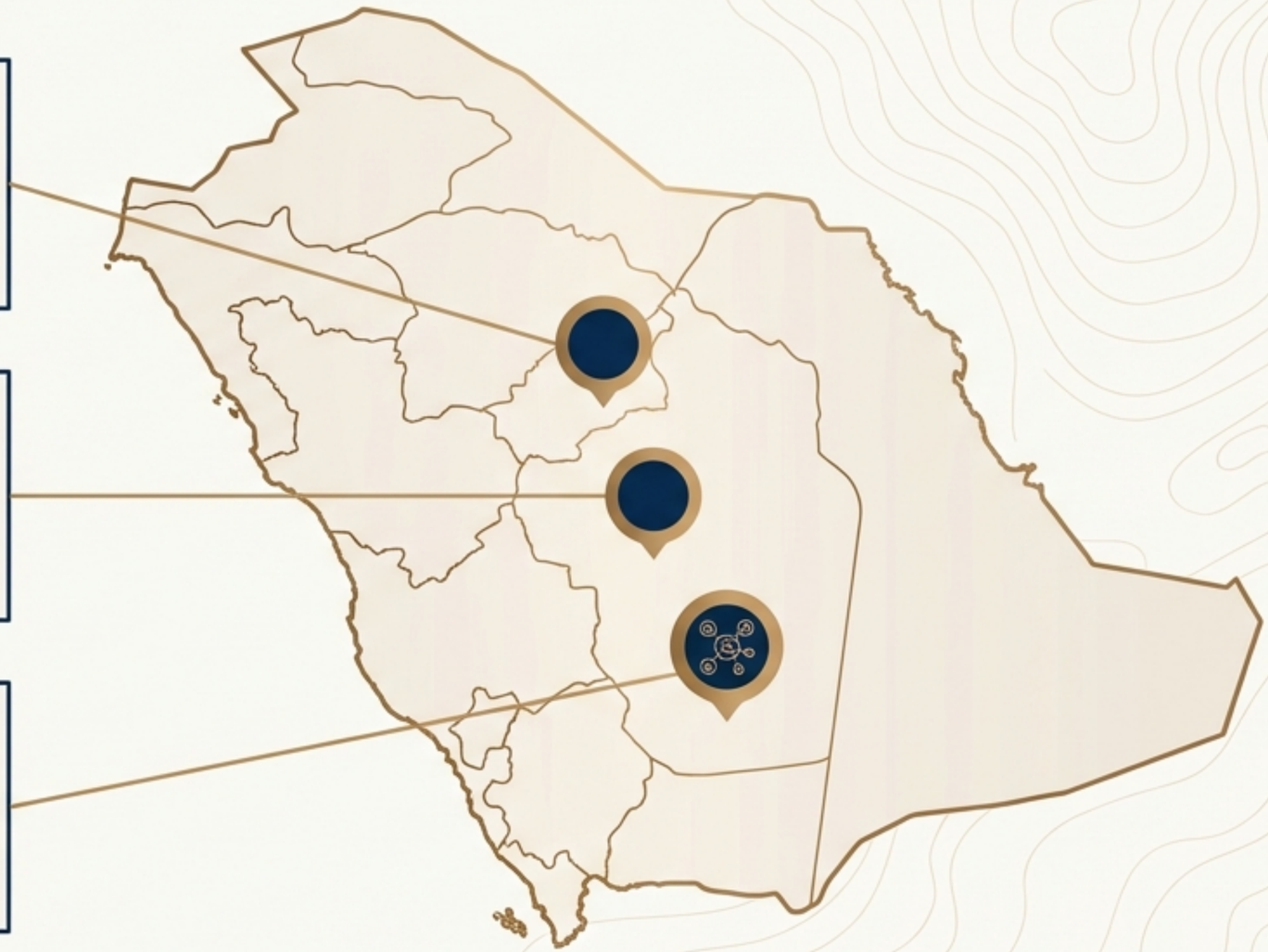
"You cannot improve what you cannot measure."

Local Context: Reliability in the Kingdom

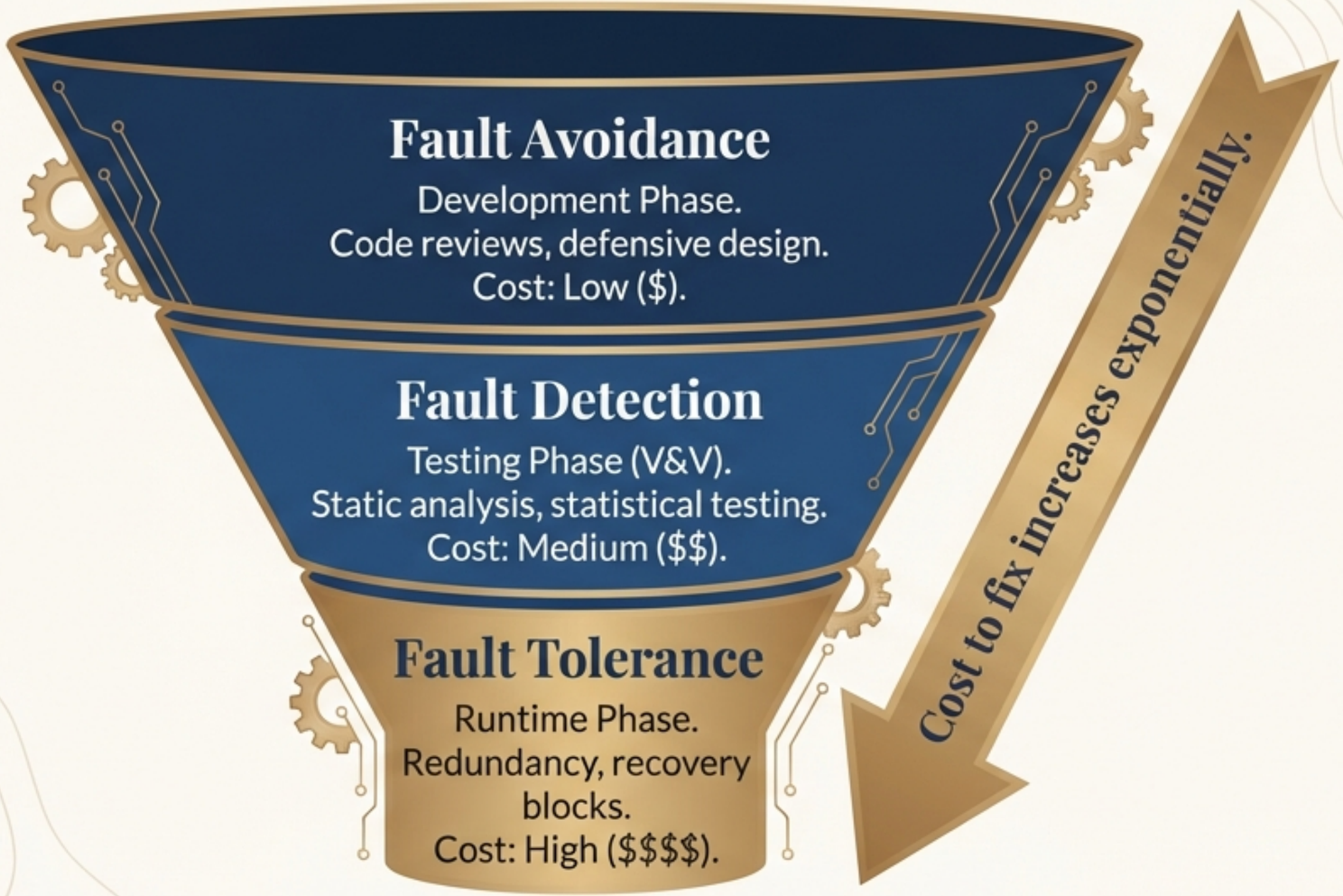
Absher (Government Services)
Metric: High Availability
Why: National productivity depends on it; millions of daily users.

Saudi Banking / SAMA
Metric: ROCOF & Integrity
Why: High transaction density; zero tolerance for balance errors.

STC (Telecom Infrastructure)
Metric: Rapid Recovery
Why: A 1-hour outage is catastrophic; rapid restoration is prioritized over perfection.

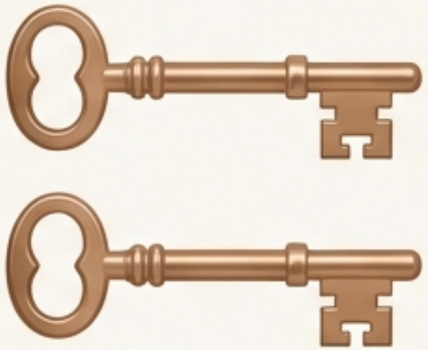


Strategy: The Three Lines of Defense



Architectural Patterns: Redundancy vs. Diversity

Redundancy



Definition: Multiple identical components running in parallel.

Purpose: Protects against hardware failure (wear and tear).

Diversity (N-Version Programming)

Definition: Different implementations of the same specification.

Purpose: Protects against design & software bugs.

Example: ATM Network Architecture uses High Availability Servers (Diversity) + Redundant Terminals.

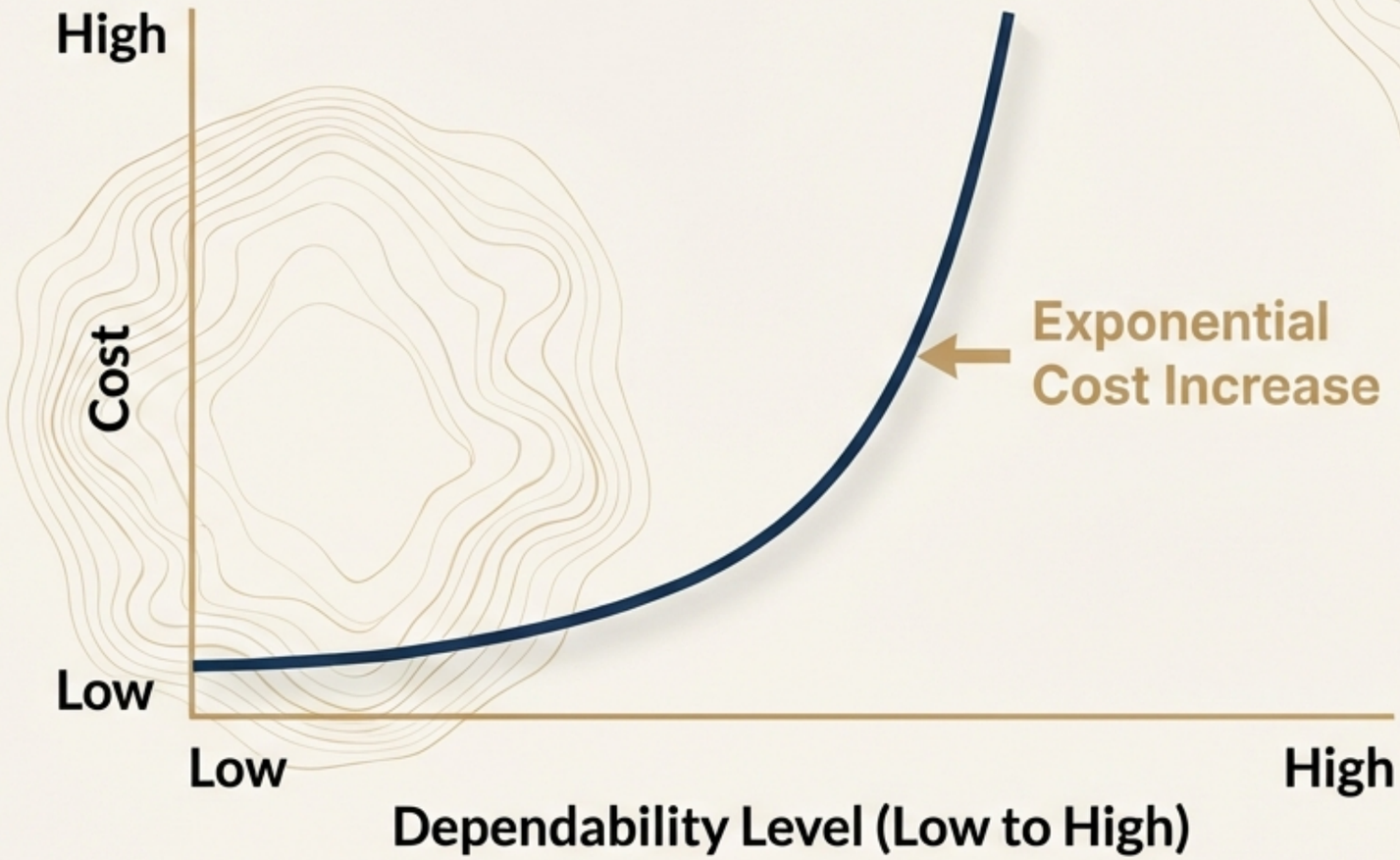
Programming for Reliability: The 8 Commandments

1. **Limit Visibility:**
   Information hiding to prevent accidental corruption.
2. **Validate All Inputs:**
   Check ranges and types. Never assume data is correct.
3. **Handle All Exceptions:**
   Catch and log everything. No silent failures.
4. **Minimize Error-Prone Constructs:**
   Avoid goto, recursion, and floating-point comparisons.
5. **Restart Capabilities:**
   Allow the system to recover state after failure.
6. **Check Array Bounds:**
   Explicitly prevent buffer overflows.
7. **Include Timeouts:**
   Assume external calls will hang.
8. **Name All Constants:**
   Avoid magic numbers for maintainability.

The Cost of Perfection

The Challenges

- **Operational Profile Uncertainty:** Testing actual user behavior.
- **High Cost of Test Data:** Generating realistic scenarios.
- **The ‘Silent’ Failure:** Wrong results without a crash.



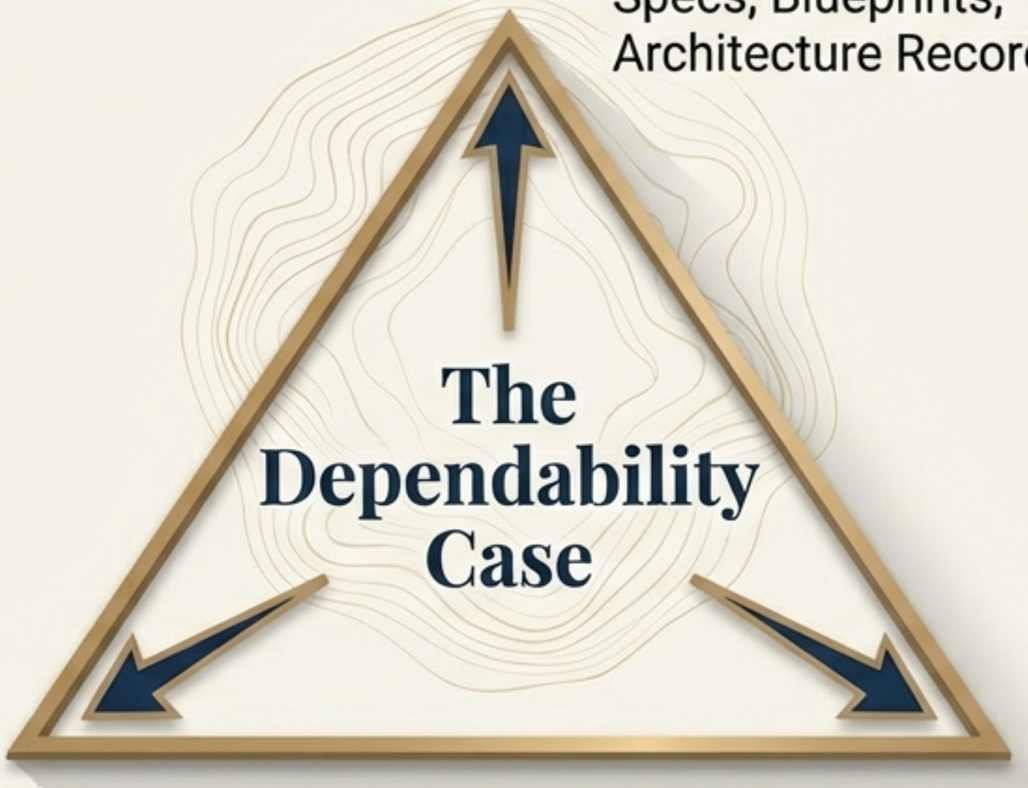
100% Reliability is infinitely expensive. Engineers must trade-off cost vs. risk.

The H-Stack: The Engineer's Mindset

"Technology fails without human discipline. Reliability bridges technical knowledge and professional wisdom."

Evidence & Triangulation

Design Evidence
Specs, Blueprints, Architecture Records.



Runtime Evidence
Incident Logs, Failure Counts.

Process Evidence
Code Reviews, V&V Logs.

Note: Screenshots ≠ Runtime Evidence unless traceable to a log.

Portfolio Task: Build Your Proof

1. **Select a Real-World System** (e.g., Hajj Management, Delivery App).
2. **Define the Profile:** Who uses it and what is the usage pattern?
3. **Choose the Metric:** Justify POFOD vs. ROCOF vs. AVAIL.
4. **Propose 1 Strategy:** Describe one architectural or code-level decision.

Deliverable: 1 Diagram + 300-word justification.

Summary & Value

Reliability is a Design Choice, not an accident.

Perceived Reliability (User Trust) > Technical Perfection.

The goal is to bridge Technical Knowledge with Professional Wisdom.

BUILD YOUR PROOF