INTRODUCTION TO FAULT TOLERANT COMPUTING

What is a Fault Tolerant System?

- A **Fault-tolerant system** is one that can continue to correctly perform its specified tasks in the presence of **hardware failures** and **software bugs**.
- Fault tolerance is the **attribute** that enables a system to **achieve fault tolerant** operation.
- Fault-tolerant computing denotes performing computing in a fault tolerant manner.

Origins of Fault-Tolerant Comp.

- **Early digital** computers made extensive use of fault-tolerant designs
  - Primarily to achieve high reliability using unreliable basic components
- As components became reliable… fault tolerance became **less of an issue**.
- With distributed computing and **highly mission critical** applications such as real-time computing: fault-tolerance is again an issue.

Goals of Fault Tolerance

- A natural question: **Why fault tolerance is important?**
- Fault tolerance is an attribute built into a system to achieve certain design goals.
- Some **requirements to be met** in this regard are
  - Reliability
  - Availability
  - Safety
  - Performability
  - Dependability
  - Maintainability
  - Testability

Reliability

- Reliability $R(t)$ of a system is a function of time defined as the conditional probability that the system will perform correctly **throughout the interval** $[t_0, t]$ given it was correct at t_0.
- Reliability is used to characterize systems where momentary failures are unacceptable as well.

Reliability…

- If repair is impossible, time interval $[t_0, t]$ can be fairly long (e.g., space application tens of years).
- Customary to use $0.999999 - 0.9_6$.
- What the **difference between fault-tolerance and reliability**?
  - Fault tolerance improves reliability
  - Do fault tolerant systems always have high reliability?

Reliability…

- In general, fault tolerant systems need **not have high reliability**.
- Likewise, highly reliable systems need **not be fault tolerant**.

Availability

- Availability is another design goal to achieve through fault tolerance.
- Availability *A(t)* is a function of time – probability that the system is operating correctly and is available to perform its functions at **time instant** *t*.
- **Availability** differs from **reliability** because the latter depends on the **interval of time**.

Availability…

- A system **can be highly available** and yet **experience frequent** but very **short periods** of **inoperability**.
- Availability depends on:
  - How often a system becomes inoperable
  - How quickly the system gets repaired and restored to the original working order
- **Availability** is important for systems that are **meant to provide service** as often as possible.

Safety

- Safety *S(t)* is the probability that a system will either perform its **functions correctly** or will **discontinue its operation without disrupting** the overall operation.
- Safety is a measure of the **fail-safe capability** of the system (e.g., fail-safe brakes, fail-safe autopilot systems in airplanes).
- Safety and reliability are **different**…

Performability

- Many cases it is desirable to design systems that continue to operate correctly after some component failure – with diminished performance levels.
- Performability *P(L,t)* of a system is the **probability** that the system **performance** will be at or **above level** *L* at the instant of time *t*.
- Performability differs from reliability -- performability is the **measure of subset** of functions performing correctly.

Maintainability

- Maintainability *M(t)* is the probability that a **failed system** can be **restored** to **operational condition within** a specified **period** *t*.
- Restoration includes **locating** the problem, physically **repairing** the system, and **bring the system back** to operation.

Testability

- Testability is the ability to **test for certain attributes** within a system. Measures of testability allow us to assess the ease with which certain tests can be performed.

Dependability

- Dependability encompasses the concepts of reliability, availability, safety, maintainability, performability, and testability.
- Dependability is a measure of ***quality of service*** a particular system provides.

Applications of FTC

- Fault-tolerance computing is used in a number of important fields. The applications of FTC can be broadly classified as:
  - **Long-life applications**: these applications require very high probability of operational at the end of a very large time interval (e.g., 10 or 20 years). Example systems include unmanned spacecrafts, storage into the future, etc.
  - **Critical-computation applications**: these applications want a system that ensures correct computations despite failures within the system. Example systems include fault-tolerant multiprocessor systems.

Applications of FTC…

- **Maintenance postponement applications**: When maintenance operations are costly or inconvenient, fault tolerance is used to postpone them. Space or remote data processing is good example. Maintenance crew perform servicing after many years and fault tolerant computing is expected to keep systems going in between.
- **High-availability applications**: Banking and e-commerce expect their services to be available most of the time (very high availabilities). Fault tolerance is used to fail-over to meet the availability requirements.

Basic Definitions

- Three fundamental concepts: faults, errors, and failure.

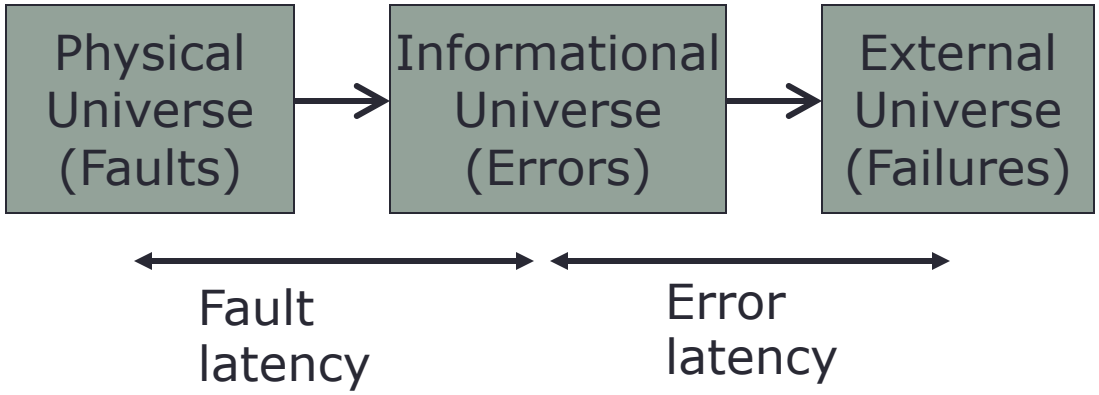
- **Fault**: physical defect, imperfection, or flaw that occurs within some hardware and software component.
- **Error**: is a manifestation of the fault. Specifically, error is a deviation from accuracy and correctness.
- **Failure**: error results in the system performing one of its functions incorrectly. Or failure is the non-performance of some action that is due or expected.

The Three-Universe Model

- **Physical universe**: this is where the faults happen. Physical universe contains the semiconductor devices, mechanical devices, printers, power supplies, software
- **Informational universe**: this is where the errors happen. Errors affect units of information such as data words within a computer. Error has occurred when some unit of information becomes incorrect.
- **External universe**: is where the user or system ultimately sees the faults or errors. I.e., where failures occur.

The Three-Universe Model

- **Cause-effect** relationships between faults, errors, and failures in the **three-universe** model.



- This model leads to two important parameters: ***Fault latency*** and ***Error latency***.
- A **latent fault** is one that is present in the system but has not produced an error!

Causes of Faults

- Problems at several points in the design process can lead to faults within a system.
- Faults can be associated with problems in four basic areas:
  - Specifications
  - Implementation
  - Components
  - External factors

Causes of Faults: Specification…

- Specification mistakes include:
  - Incorrect algorithms, architectures, or hardware and software design specifications
- Example, designer of a digital circuit incorrectly specified the timing characteristics of some of the circuit components
- Design could perform correctly under some conditions

Causes of Faults: Implementation..

- Implementation is defined as the process is turning specifications into concrete implementations.
- Implementation mistakes include:
  - Faults due to poor design
  - Poor component selection
  - Poor construction
  - Or software coding mistakes
- System can be working correctly for some input conditions.

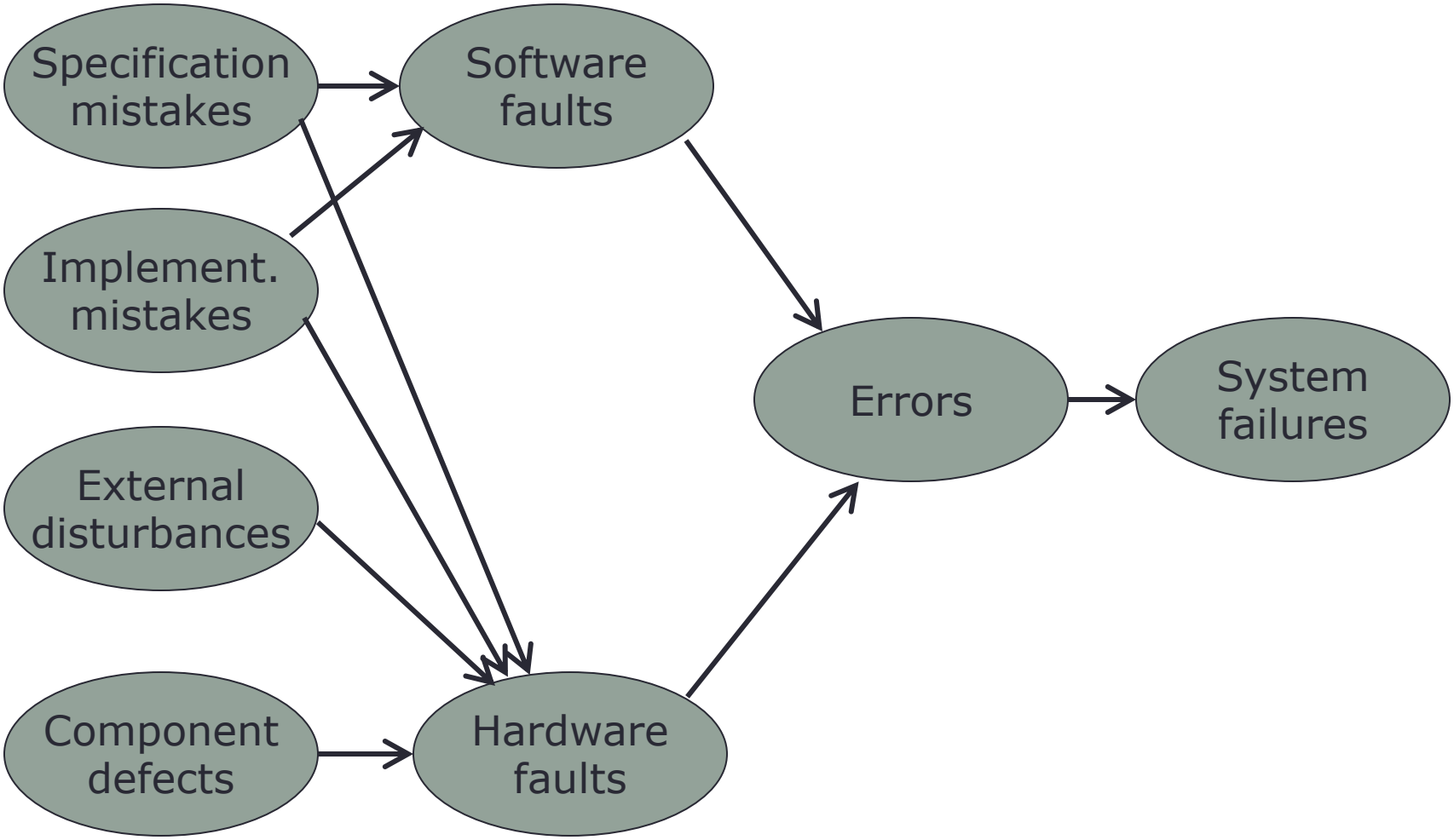
Causes of Faults: Component..

- Manufacturing imperfections fall into this class.
- As well, component wear-out

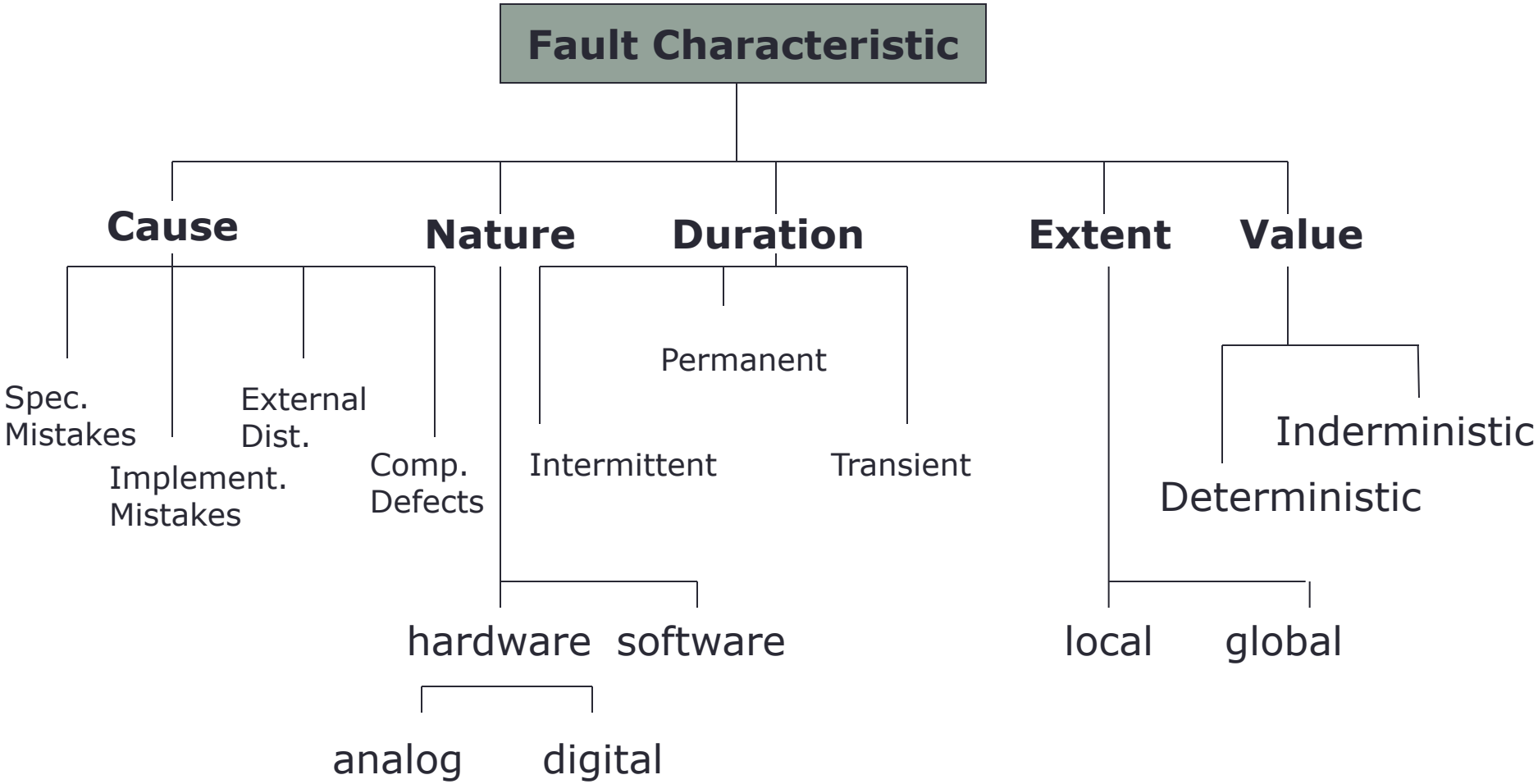
Causes of Faults: External..

- External disturbances:
- For example, radiation, electromagnetic interference, battle damage, operator mistake, and environmental extremes.

Causes of Faults: Summary



Characteristics of Faults



Fault Models

- Two basic models:
  - Logical Stuck at 1 or 0 model. Usually used in the digital system modeling.
  - Transistor stuck-fault model.
- We will consider more models with the case studies

Design Philosophies

- Primary techniques:
  - **Fault avoidance**: techniques to prevent faults in the first place. This can include design reviews, component screening, testing, and other quality control methods.
  - **Fault masking**: techniques to prevent faults from introducing errors into the informational structure of the system. Error correcting memories are one example.

Design Philosophies…

- **Fault tolerance**: is the ability of a system to continue despite faults. Fault tolerance can be achieve in different ways. Fault masking is one approach. Another is to detect, locate, and remove faults (possibly by reconfiguring the system).
- To implement the above technique we follow: detect, locate, and recovery.
- **Fault containment**: is to confine faults to localized environments and prevent it from propagating throughout a system.

Design Philosophies…