# Dependable Systems

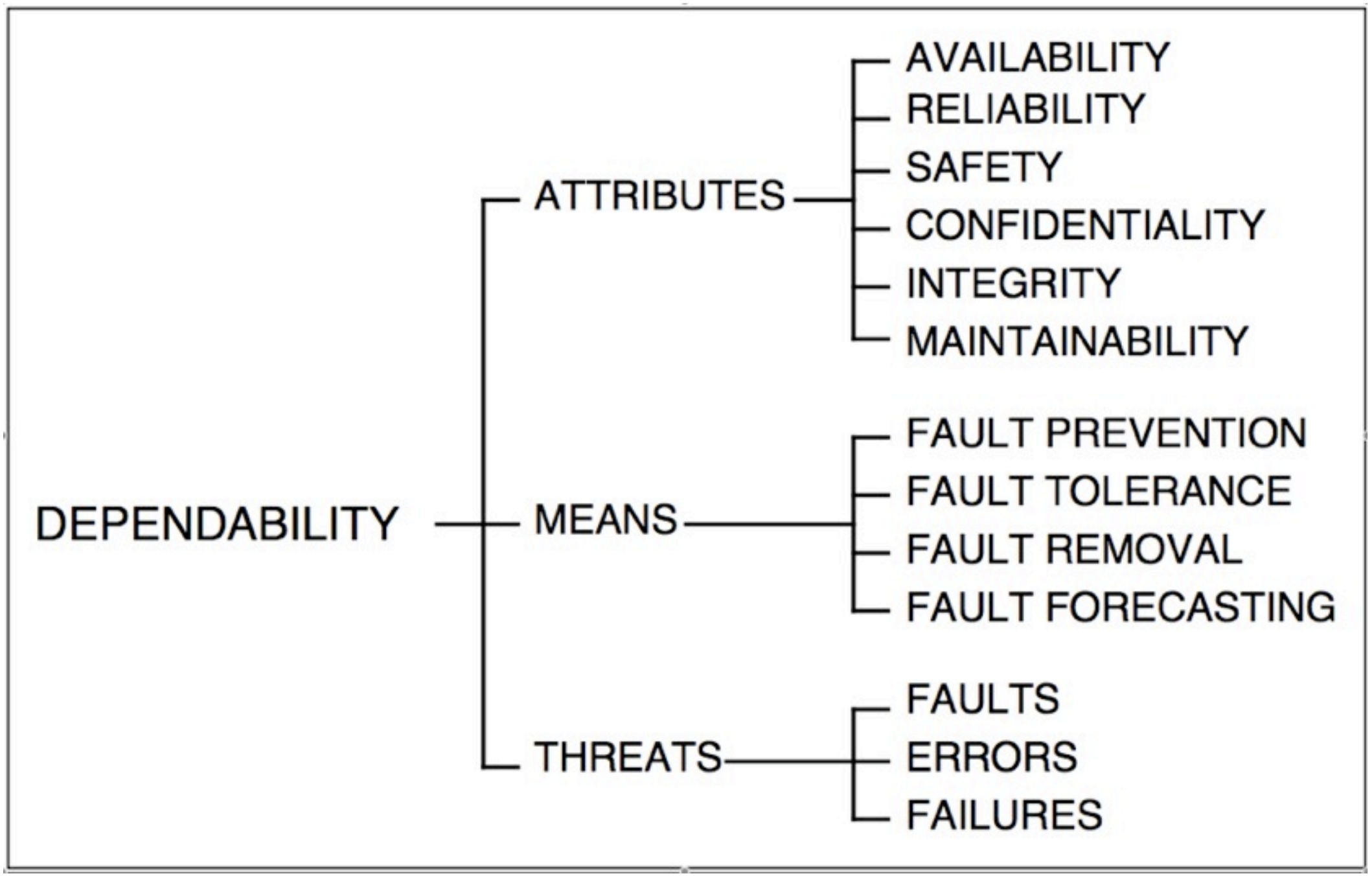
# Definitions and Metrics (II)

Dr. Peter Tröger

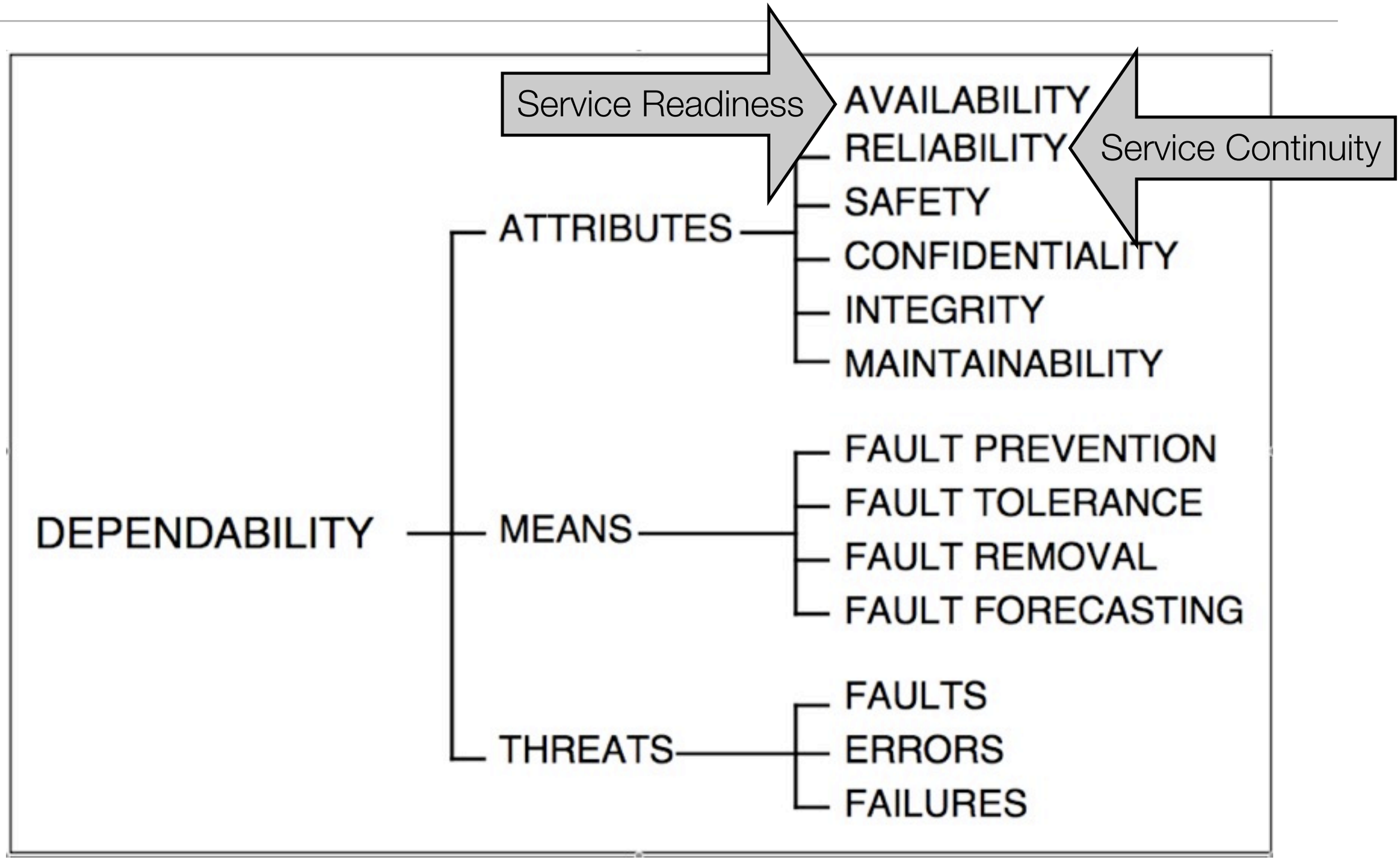
Sources:

J.C. Laprie et al.. Dependability: Basic Concepts and Terminology

# Dependability Tree (Laprie)

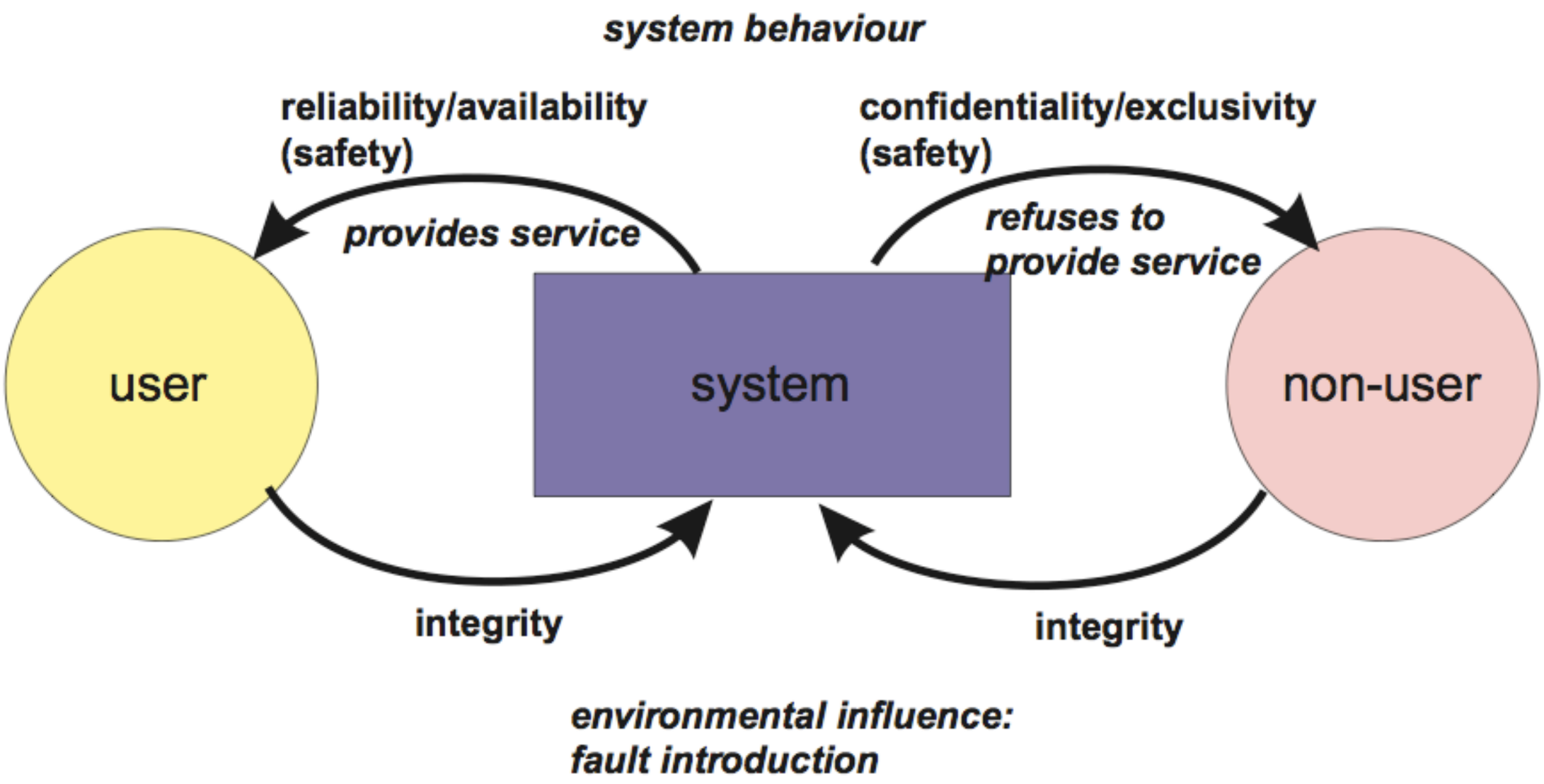


# Dependability Tree (Laprie)

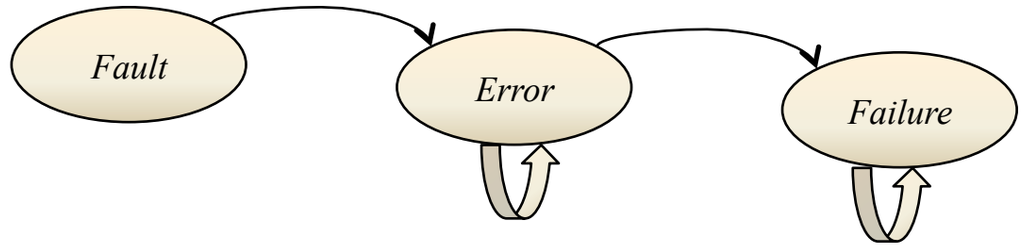


# Unified System model (Jonsson)

- Fault introduction can happen from different sides
- Different viewpoints must be considered

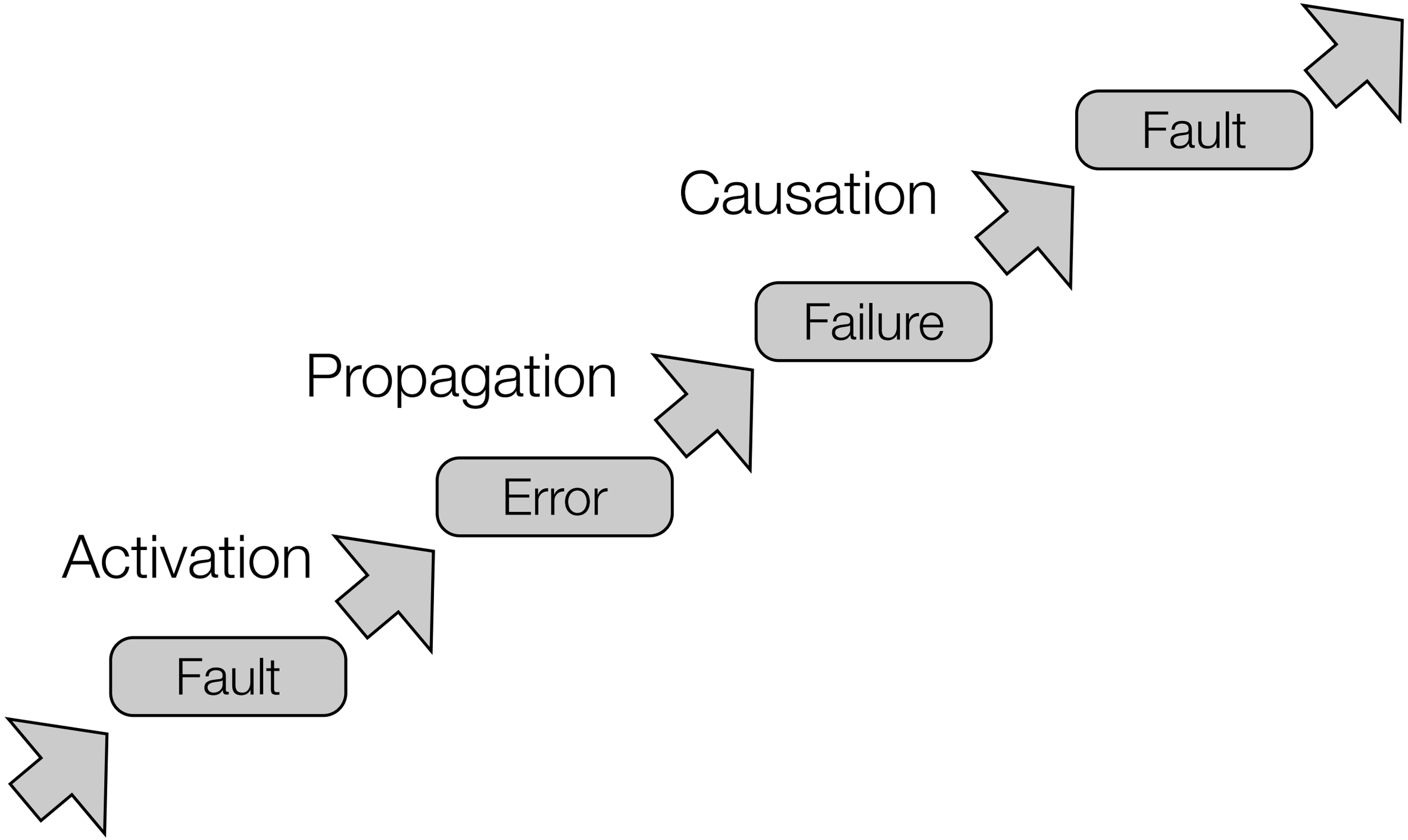


# Impairments to Dependability

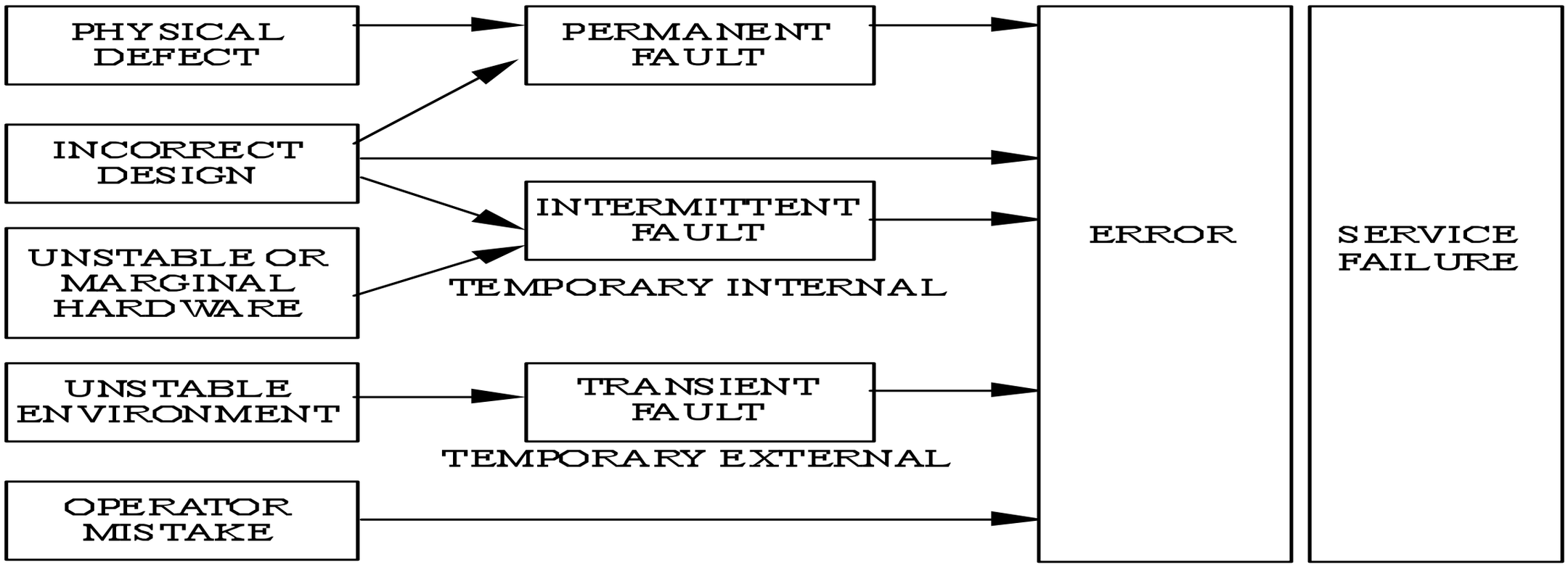



- System **failure - ‚Ausfall‘**
  - Event that occurs when the service no longer complies with the specification / deviates from the correct service.
- System **error - ‚Fehler(zustand)‘**
  - Part of system state that can lead to subsequent failure
- System **fault - ‚Fehlerursache‘**
  - Adjudged or hypothesized cause of an error
- Failure occurs when error state alters the provided service
- Systems are build from connected components, which are again systems
- Fault is the consequence of a failure of some other system to deliver its service

# Chain of Dependability Threats (Avizienis)



# Chain of Dependability Threats




SOURCES OF ERRORS

[from Siewiorek and Swarz]

# Faults

- High diversity in possible sources and types
  - Fault nature
    - **Accidental** faults (‚Zufallsfehler‘) vs**. intentional** faults (‚Absichtsfehler‘)
  - Fault origin viewpoints
    - Phenomenological causes: **Physical** faults vs**. human-made** faults
    - System boundaries: **Internal** faults vs**. external** faults
    - Phase of creation: **Design** faults vs**. operational** faults
- Temporal persistence
  - **Permanent** faults vs**. temporary** faults

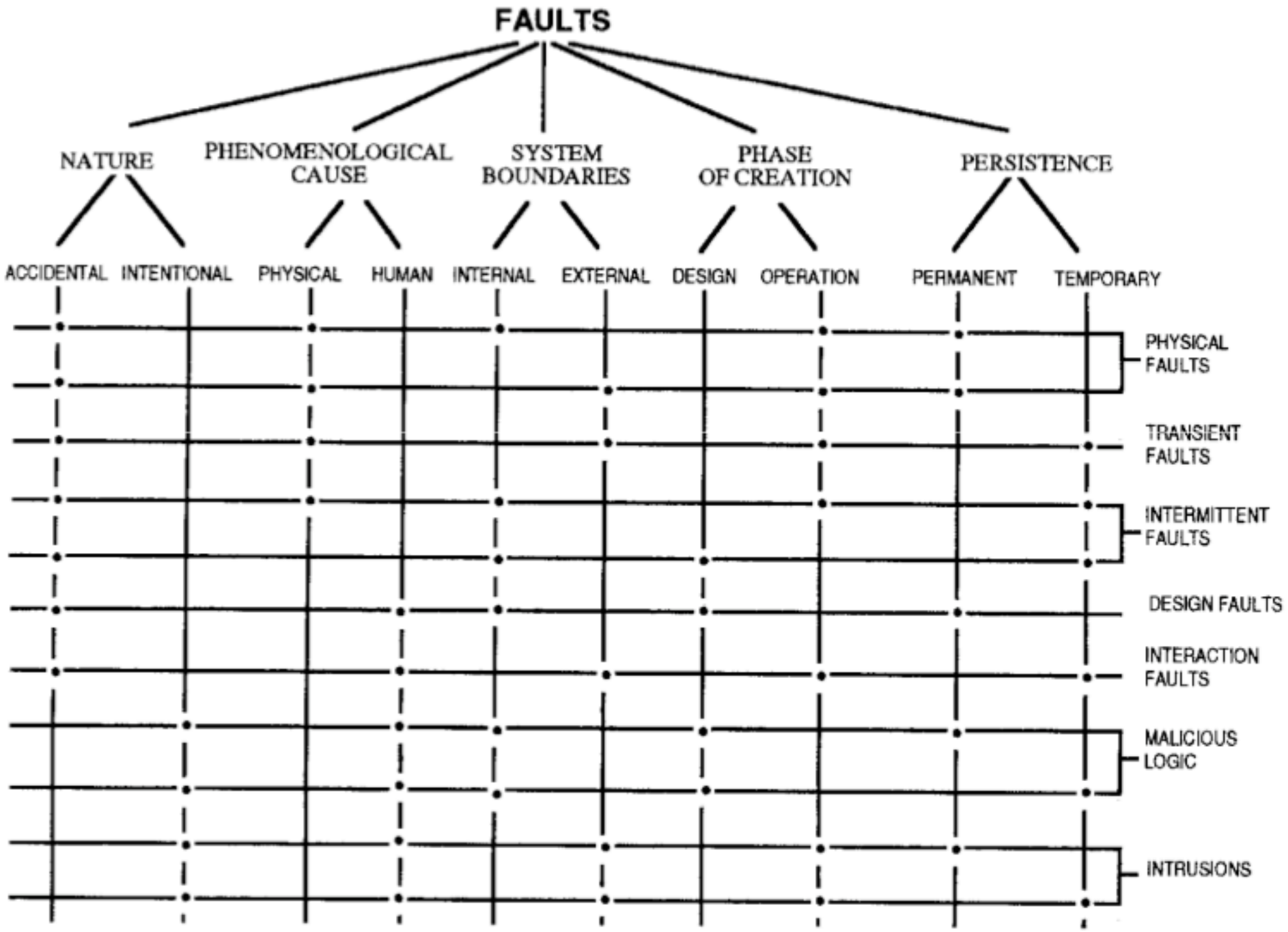
# Observations on Faults

- An external fault is a design fault - inability or refusal to foresee all situations
- Design faults are created during system development, system modification, or operational procedure creation and establishment
- Just replacing broken version of the same component leads to **recurrent faults**
- Physical faults are **accidental faults**
- Temporary external accidental physical faults are also called **transient faults**
- Temporary internal accidental faults are also called **intermittent faults**
  - Examples: Pattern-sensitive memory hardware, system overload
  - Arbitrary concept - Permanent faults with unknown activation condition
- Intentional and design faults are human-made faults

# Observations on Faults

- A fault is **active** when it produces an error
- A non-active internal fault is a **dormant fault** (‚inaktive Fehlerursache‘)
  - Origin in hardware fault analysis - often cycling between dormant and active
- Many specialized versions of the term ‚fault‘, e.g. **bug**
  - **Heisenbug** - Intermittent software fault, **Bohrbug** - Permanent software fault
  - **Mandelbugs** - Appear chaotic due to many dependencies
- **Fault-tolerant system design** is a contradiction
  - Design demands specification, faults are non-specified cases
  - Solution: Specification for fault-free case + additional fault specification
- Fault can mean performance or timing faults (derivation from expected load / timing)

# Fault Characterization (Laprie & Kanoun)

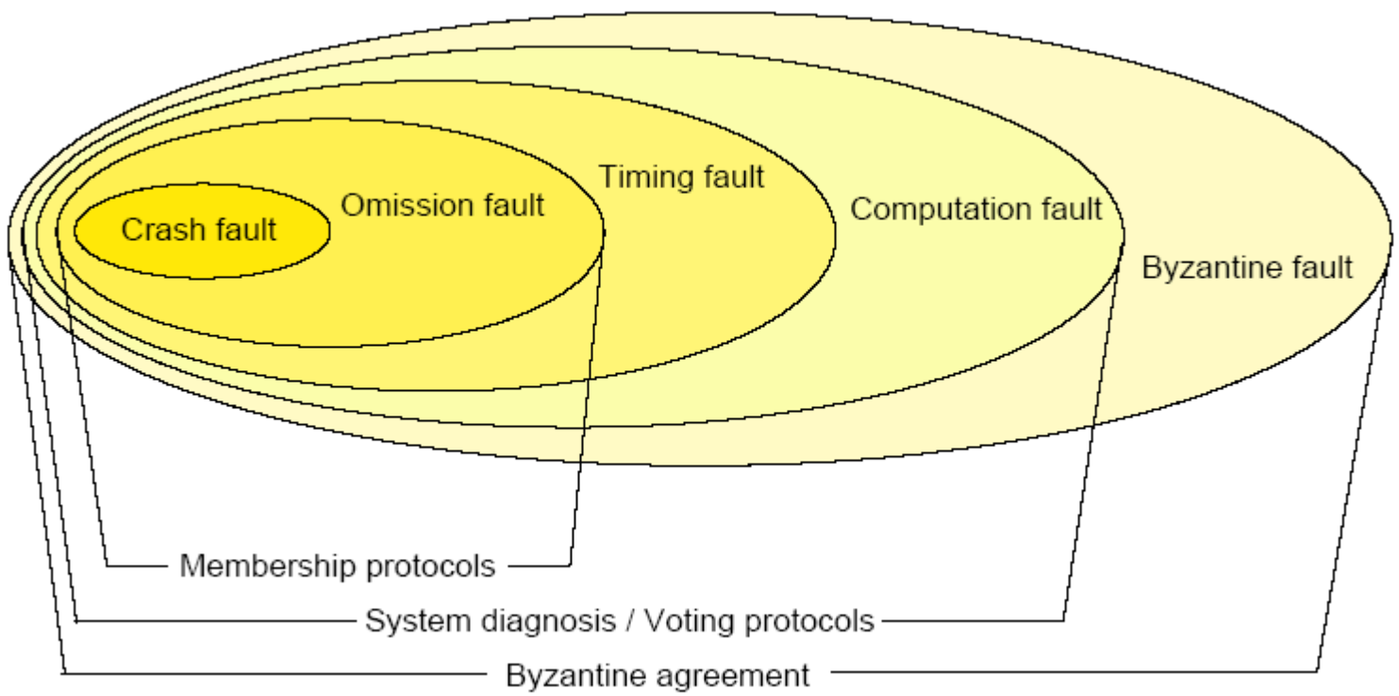


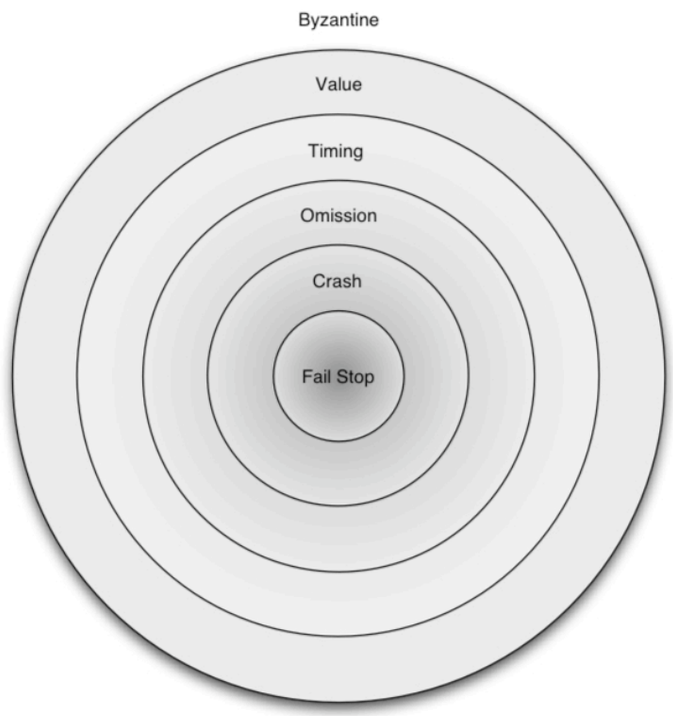
# Fault Model

- Faults can be classified into different categories on different abstraction levels
  - Physics (unusual)
  - Circuit level / switching circuit level
    - Interesting for hardware design research (not this course)
      - Investigate logical signals on connections
        - stuck-at-zero, stuck-at-one, bridging faults, stuck-open
  - Register transfer level
  - Processor-memory-switch (PMS) level
  - System level

# Fault Model

- Idea from hardware background, meanwhile also in software
  - Usage: How many faults of different classes can occur ? What do I tolerate ?
- Process as black box, only look on input and output messages
- Link faults are mapped to the participating components
- Timing of faults: Fault delay, repeat time, recovery time, reboot time, ...

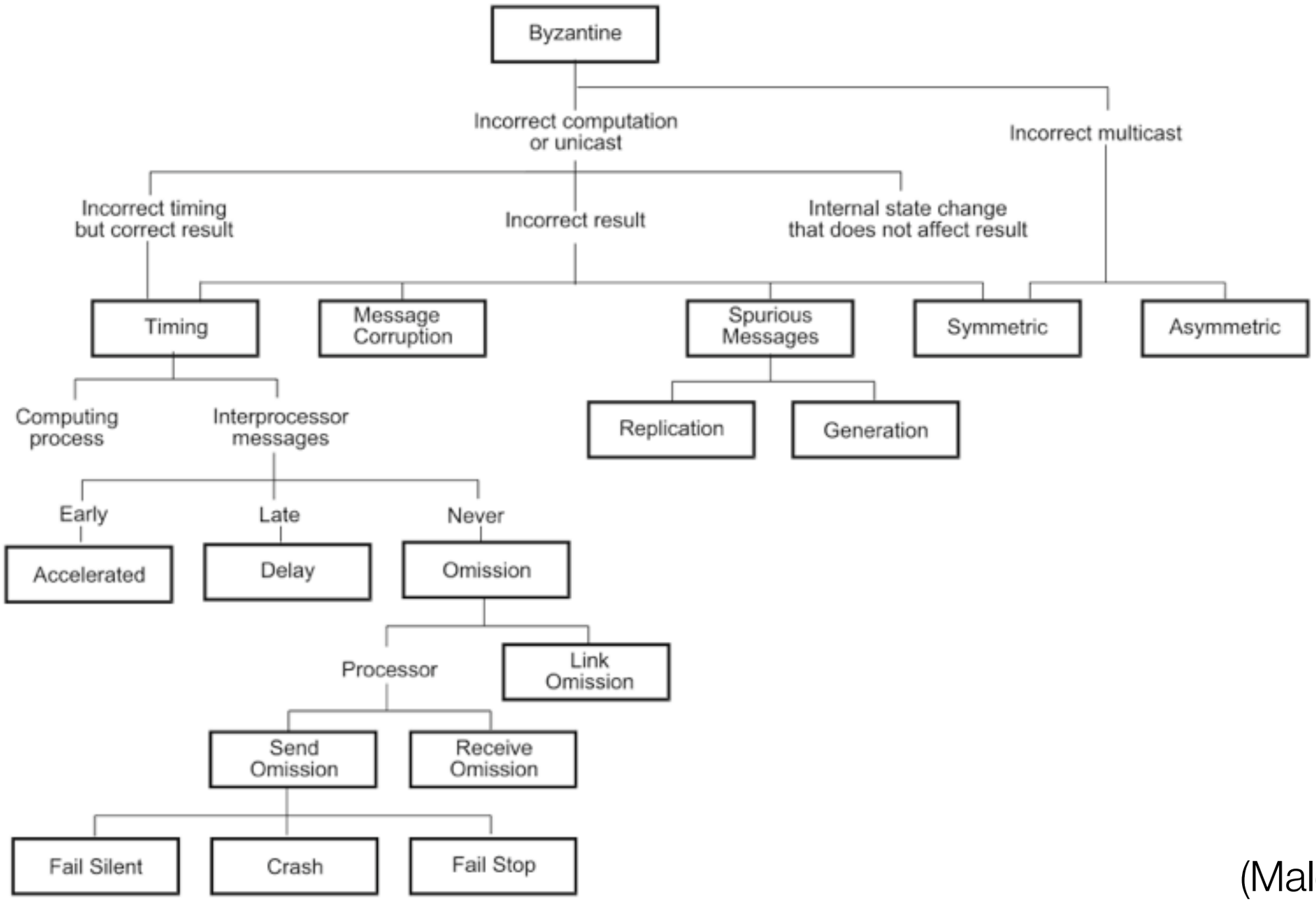


# Fault Model

- **Fail-Stop** Fault : Processor stops all operations, notifies the other ones
- **Crash** Fault : Processor looses internal state or stops without notification
- **Omission** Fault : Processor will break a deadline or does not react to some task
  - Send / Receiver Omission Fault: Necessary message was not not sent / not received in time
- **Timing** Fault / **Performance** Fault : Processor stops / reacts to a task before its time window, after its time window, or never
- **Incorrect Computation** Fault : No correct output on correct input
- **Byzantine** Fault / **Arbitrary** Fault : Every possible fault
  - Authenticated Byzantine Fault : Every possible fault, but authenticated messages cannot be tampered

# Fault Hierarchy in Distributed Systems

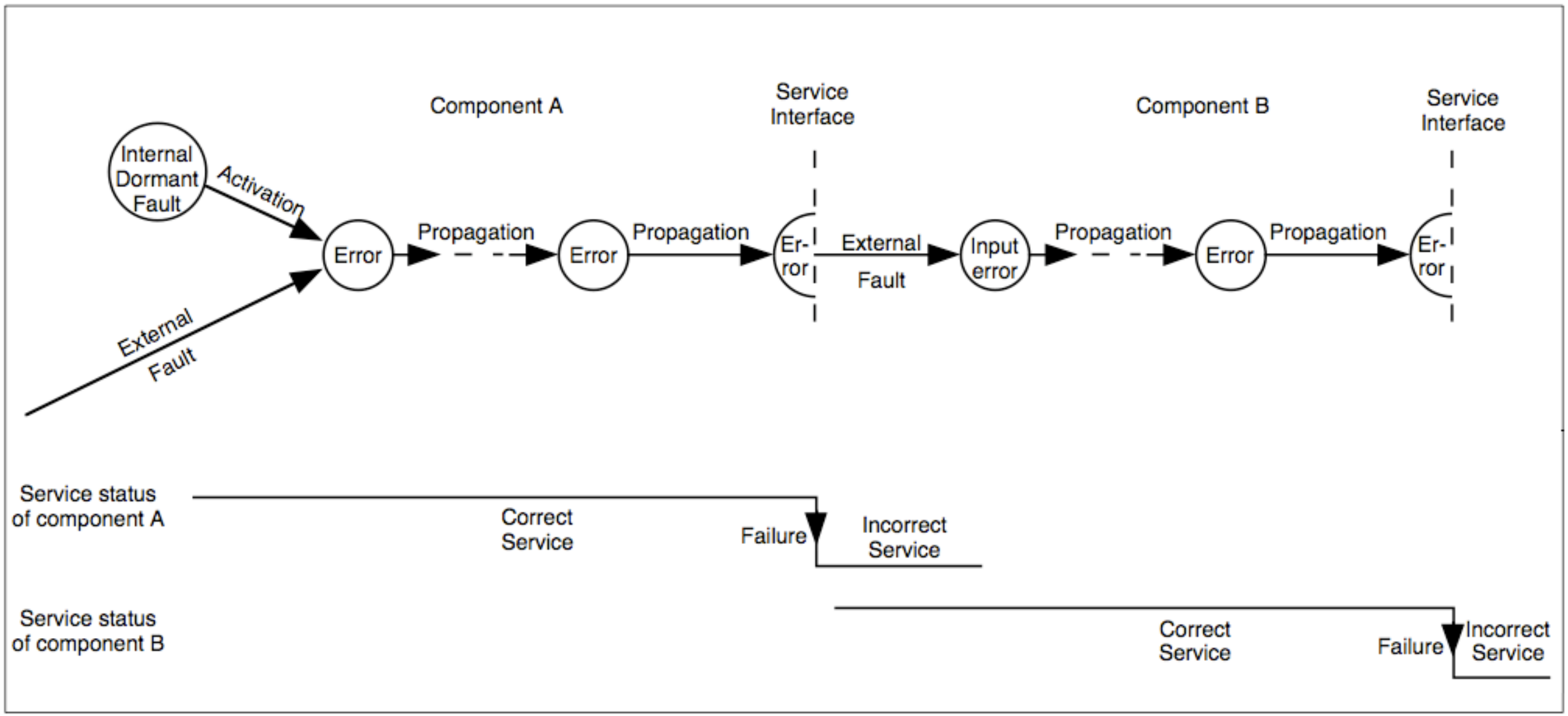



(Malek)

# Errors

- State of the system, not an event !
- Escalates to failure depending on
  - Intentional / unintentional redundancy
  - System activity
  - User‘s definition of a failure
    - Examples: Maximum outage time, acceptable delay, retransmission rate
- **Latent** (not recognized) vs**. detected** error coming from an active fault

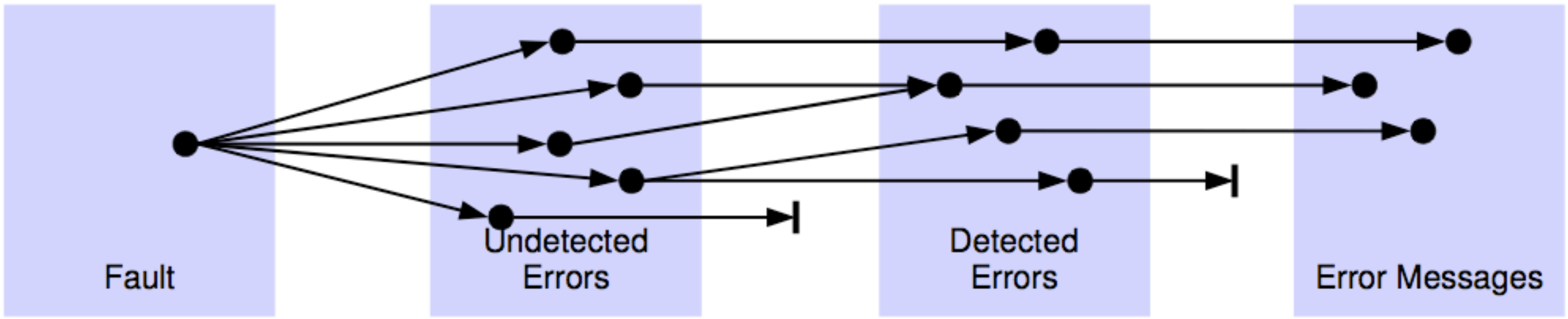
# Error Propagation




(C) Avizienis

# Error Message Occurrence (Hansen & Siewiorek)

- Same fault can lead to different errors
- N:M relationship of undetected errors and detectors
- Detected errors might not be logged



# Failures

- Non-compliance with the specification - **arbitrary failure** (‘willkürlicher Ausfall‘)
- System failures can be further categorized in **failure modes**
  - **Fail-silent** / **crash failure** mode - incorrect results are not delivered
  - **Fail-stop** mode - constant value is delivered
- Failure mode view points
  - Failure mode **domain** - what is influenced
    - Service result - **value failures,** service timeliness - **timing failures**
    - Service availability - **stopping failures**
  - **User perception** in this mode - consistent / inconsistent for all users
  - Failure **consequences** in this mode - allow ordering of failure modes

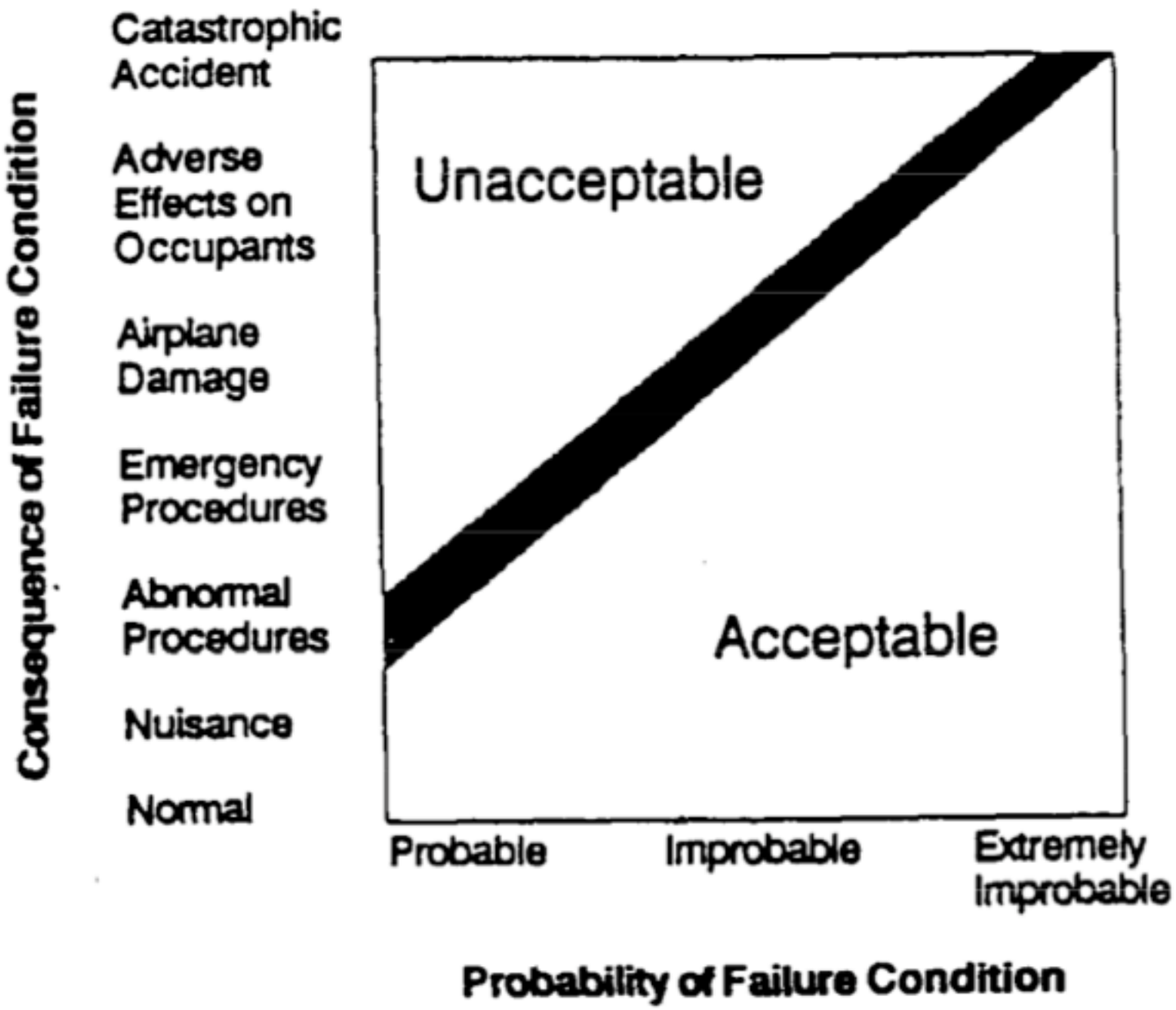
# Failure Severity (‚Schweregrad des Ausfalls‘)

- **Benign failures** (‚unkritische Ausfälle‘)
  - Failure costs and operational benefits are similar
  - Sometimes also umbrella term for failures only detected by inspection
  - A system with only such failures is **fail-safe**
- **Catastrophic failures** (‚kritische Ausfälle‘)
  - Costs of failure consequences are much larger than service benefit
- Grading of failure consequences on overal system depends on application
  - Inoperation of flying airplane - Catastrophic stopping failure
  - Inoperation of train - Benign stopping failure
- **Criticality** - Highest severity of possible failure modes in the system

# Criticality Levels Example: DO-178B Standard

- *Software Considerations in Airborne Systems and Equipment Certification*
  - Mature document, developed for more than 20 years
- Definition of **severity of failure conditions** for airplane, crew, and passengers
  - *Catastrophic* - Loss of ability to continued safe flight and landing
  - *Major* - Reduced airplane or crew capability to cope with operating conditions
    - Reduction in safety margins and functional capabilities
    - Higher workload or physical distress for the crew
  - *Minor* - Not significantly reduced airplane safety, slight increase in workload
  - *No effect* - Failure results in no loss of operational capabilities and no increase in crew workload

# Example: DO-178B Standard

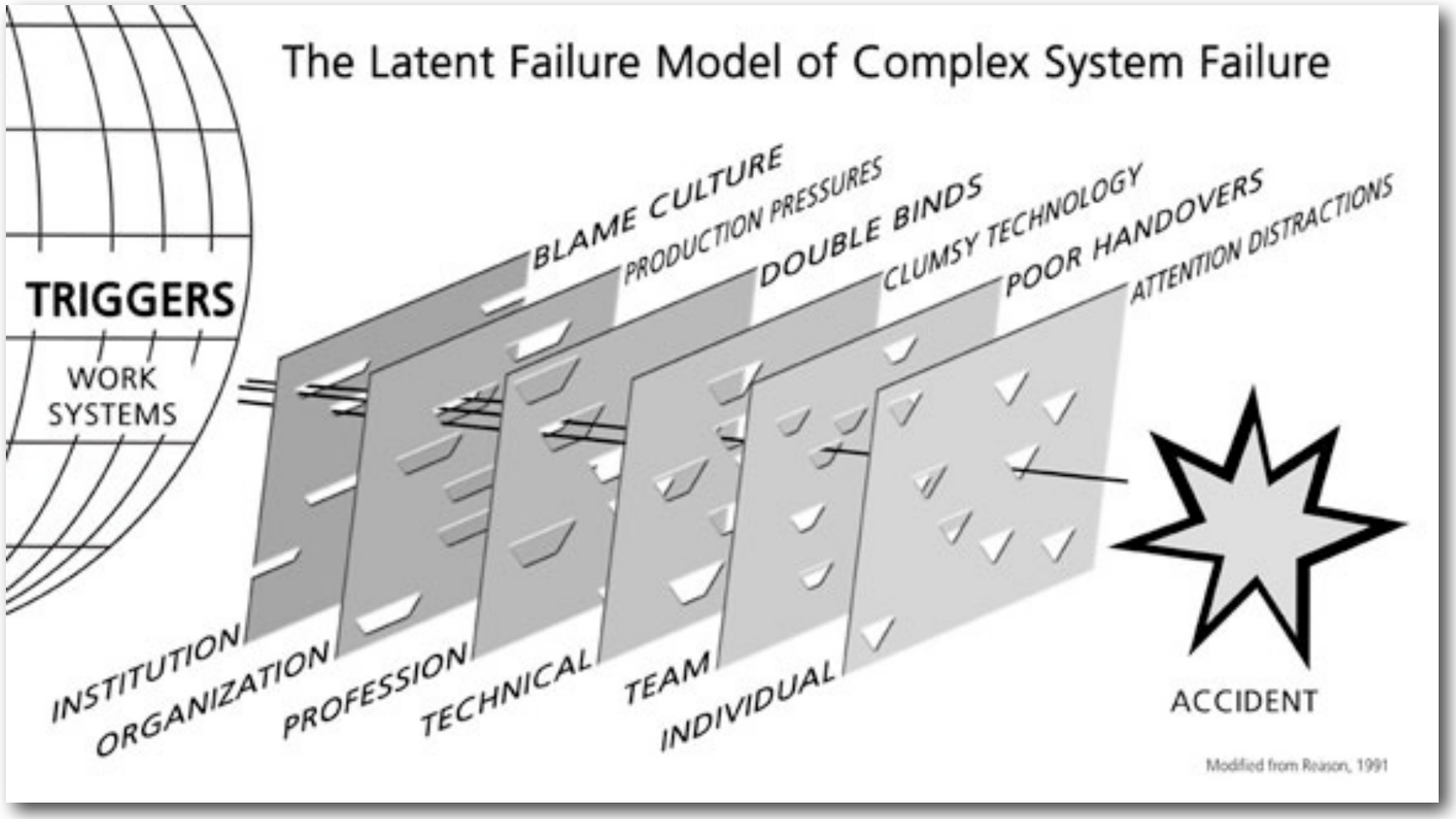


# Failure Types

- Duration of the failure
  - **Permanent** failures - no possibility fo repairing or replacing
  - **Recoverable** failures - back in operation after a fault is recovered
  - **Transient** failures - short duration, no major recovery action
- Effect of the failure
  - **Functional** failures - system does not operate according to its specification
  - **Performance** failures - performance or SLA specifications not met
- Scope of the failure
  - **Partial** failure - only parts of the system become unavailable
  - **Total** failure - all services go down

# Swiss Cheese Model (Prof. Reason)

- Origins in medical research
- Defences, barriers, and safeguards might be penetrated by fault trajectory




(C) Fernando Bernal

# Observations on Failures

- Failures vs. Load
  - Typically positive correlation
    - Increasing load can lead to wear-out - increasing failure rate
    - Higher load can show up failure causes
    - Detected faults lead to recovery activities - load increases
  - Feedback effects possible
- Related faults (attributed to a common cause) can lead to **common-mode failures**