BASIC CONCEPTS AND TAXONOMY OF DEPENDABLE AND SECURE COMPUTING

by Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr,

IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 1, NO. 1, JANUARY-MARCH 2004

BASICS

- We have discussed the basic issues of dependable systems before.
- Now we will focus more on survivability-related issues of the aforementioned paper
- Most of the material is <u>directly taken from the paper</u> and (to avoid visual clutter) will not be explicitly cited!

2) BASIC CONCEPTS

- System
  - **entity** that interacts with other **entities**
  - includes hardware, software, humans, physical world with its natural phenomena
  - **system boundary**
  - **function** is what it should do, often is described by functional specification in terms of functionality and performance
  - **behavior** is what system does to implement its functions
  - behavior is described by sequence of **states**

2) BASIC CONCEPTS

- **Total State** of a System defined by following:
  - computation
  - communication
  - stored information
  - interconnection
  - physical condition

2) BASIC CONCEPTS

- **Structure of a system**
  - set of **components** that interact
  - each component is another system
    - recursive definition
    - stops with atomic component
      - i.e., no need or not possible to further break down

2) BASIC CONCEPTS

- **Service** delivered by a system
  - in its role as **provider**
  - **user** is another system receiving service from the provider
  - **service interface** is the boundary where service delivery takes place
  - user sees **external state** of provider; remaining part is **internal state**
  - user receives service at **use interface**

2) BASIC CONCEPTS

- Threats to Dependability and Security
  - **Service failure**, or just **failure**
    - delivered service deviates from correct service
    - **transition** from correct to incorrect service

2) BASIC CONCEPTS

- Threats to Dependability and Security
  - **Service outage**
    - period of delivery of incorrect service
  - **Service restoration**
    - transition from incorrect to correct service
  - deviation from correct service may assume different forms: **service failure modes**

2) BASIC CONCEPTS

- Failure, error, fault
  - Service is sequence of system's external states
  - Service failure means ∃ at least one external state of the system that deviates from the correct service state
  - That deviation is called an **error**
  - The cause of the error is called **fault**

- Faults
  - internal fault or external
  - **vulnerability**, i.e., an internal fault that enables an external fault to harm the system, is necessary for an external fault to cause an error and possibly subsequent failure

2) BASIC CONCEPTS

- typically: **fault** causes **error**, which can cause **failure**
    - fault is **active** when it causes an error
    - otherwise it is **dormant**

2) BASIC CONCEPTS

- If functional specification of a system includes a set of several functions, then
  - failure of one or more services that implement the function may leave system in a **degraded mode**
    - still offers subset of needed services
    - e.g., slower, limited service, emergency service
    - system is said to have suffered **partial failure**

2) BASIC CONCEPTS

- Dependability Security and their Attributes
  - original definition of **dependability**
    - “ability to deliver service that can justifiably be trusted”
  - alternate definition
    - “ability to avoid service failures that are more frequent and more severe than is acceptable”

2) BASIC CONCEPTS

- Trust
  - dependence of system A on system B represents the extend to which system A's dependability is affected by that of system B
  - concept of dependence leads to that of **trust**,
    - **trust = accepted dependence**

2) BASIC CONCEPTS

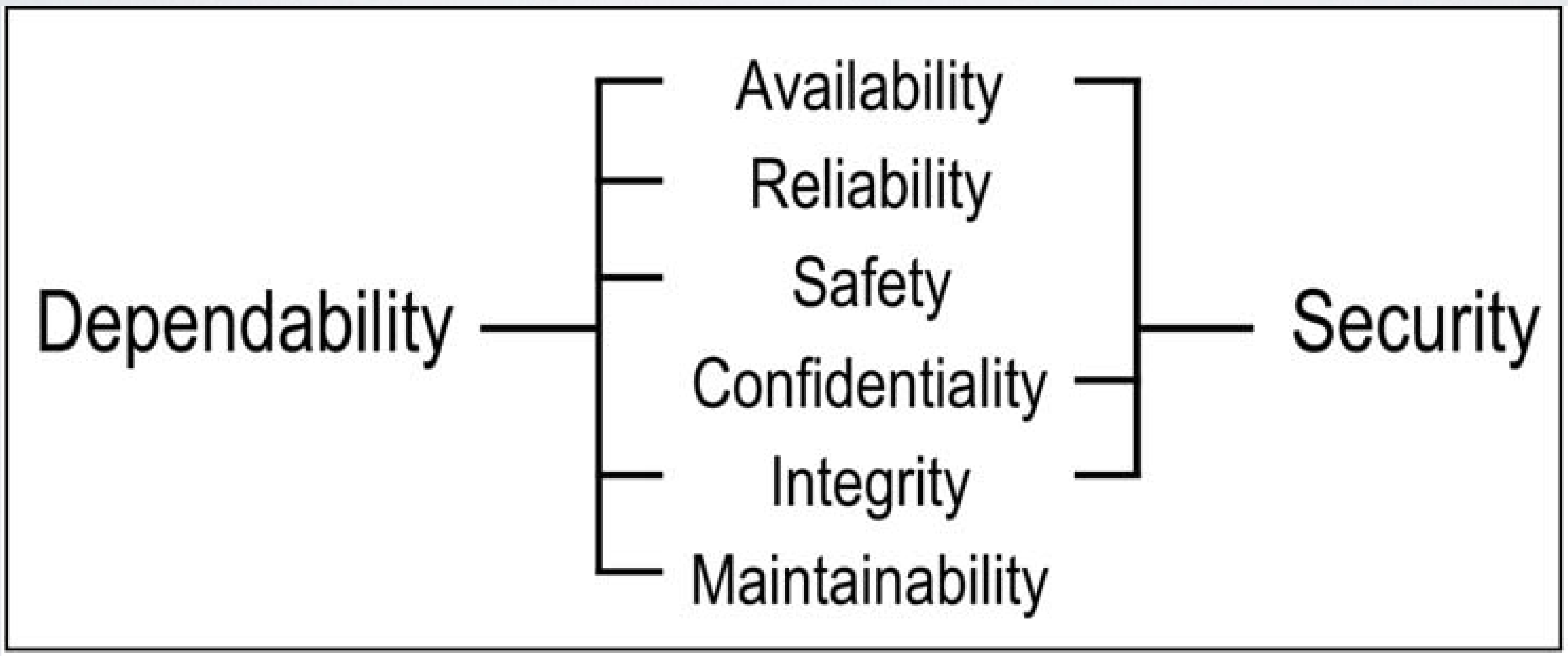
- Dependability encompasses the following attributes
  - **availability**: readiness for correct service.
  - **reliability**: continuity of correct service.
  - **safety**: absence of catastrophic consequences on the user(s) and the environment.
  - **integrity**: absence of improper system alterations.
  - **maintainability**: ability to undergo modifications and repairs.

2) BASIC CONCEPTS

- when addressing security we add
  - **confidentiality**, the absence of unauthorized disclosure of information
  - Security is composite of the attributes
    - confidentiality
    - integrity
    - availability

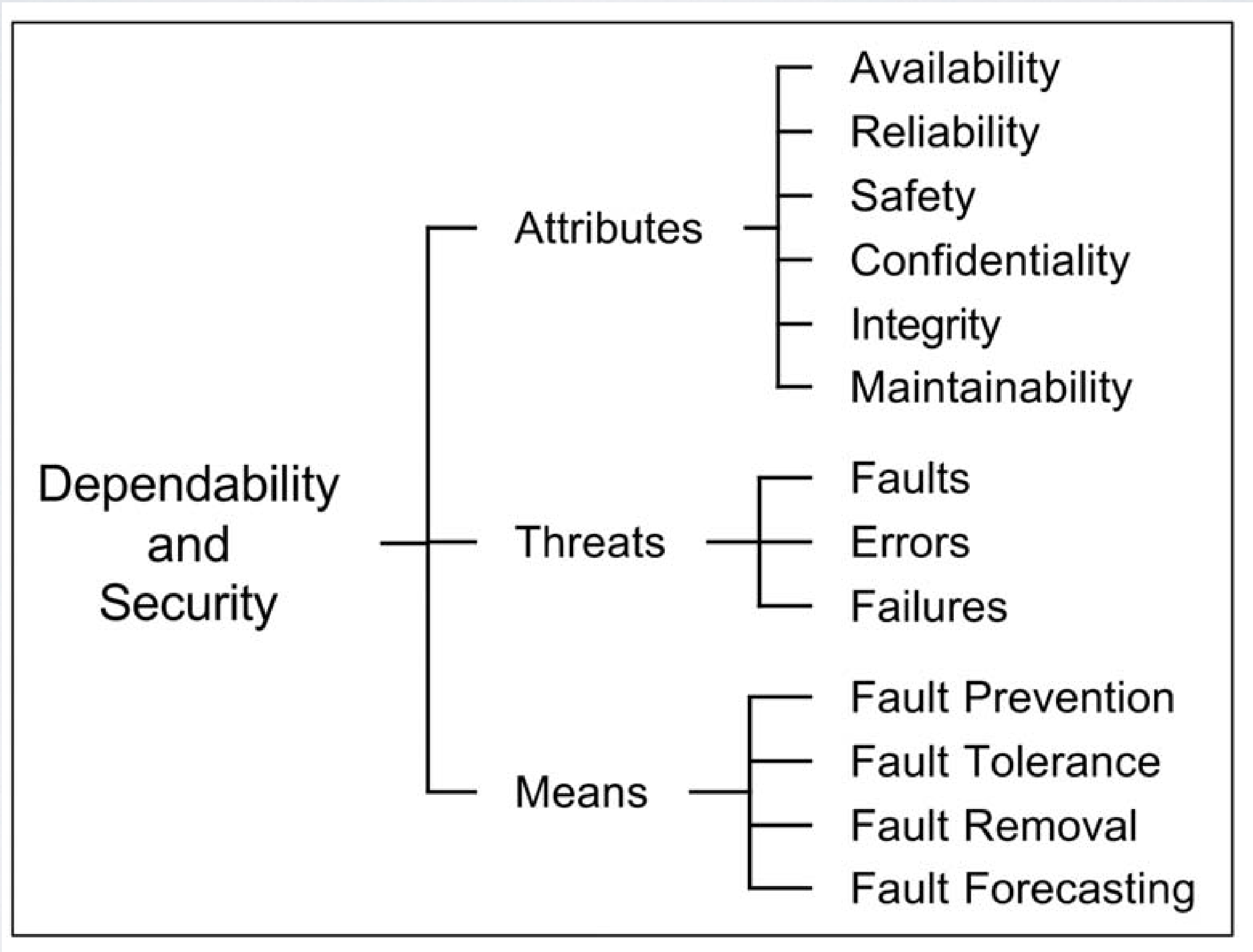
2) BASIC CONCEPTS

- Dependability and security attributes



2) BASIC CONCEPTS

- Dependability and security tree



2) BASIC CONCEPTS

- Means to attain dependability and security:
  - **Fault prevention**: prevent the occurrence or introduction of faults.
  - **Fault tolerance**: avoid service failures in the presence of faults.
  - **Fault removal**: reduce the number and severity of faults.
  - **Fault forecasting**: estimate the present number, the future incidence, and the likely con-sequences of faults.

3) THREATS TO DEPENDABILITY AND SECURITY

- 3.1: System Life Cycle: Phases and Environment
- Development phase: all activities from initial concept to green light
  - **Development Environment** of system consists of
    - **physical world** with its natural phenomena
    - **human developers** (+lacking competence, malicious objective)
    - **development tools:** software and hardware
    - **production and test facilities**

3) THREATS TO DEPENDABILITY AND SECURITY

- **Use phase**
  - System is accepted for use and starts delivering services.
  - Alternating periods of:

    Service delivery

    Service outage

    Service shutdown

- Maintenance may take place during all three periods of use phase

USE ENVIRONMENT ELEMENTS:

- **Physical world**: with its natural phenomena
- **Administrators** (includes maintainers): have authority to manage, modify, repair and use system. Some authorized humans may lack competence of have malicious objectives

USE ENVIRONMENT ELEMENTS:

- **Users**: humans or other system that receive services
- **Providers:** humans or other systems that deliver services
- **Infrastructure:** entities that provide services to the system, e.g., information sources (time, GPS) communications equipment/links, power, cooling etc.

USE ENVIRONMENT ELEMENTS:

- **Intruders:** malicious entities (human or other systems)
  - attempt to exceed authority they have
  - alter services
  - halt them
  - alter system's functionality or performance
  - access confidential information
  - examples: hackers, vandals, corrupt insiders, governments, malicious software

MAINTENANCE

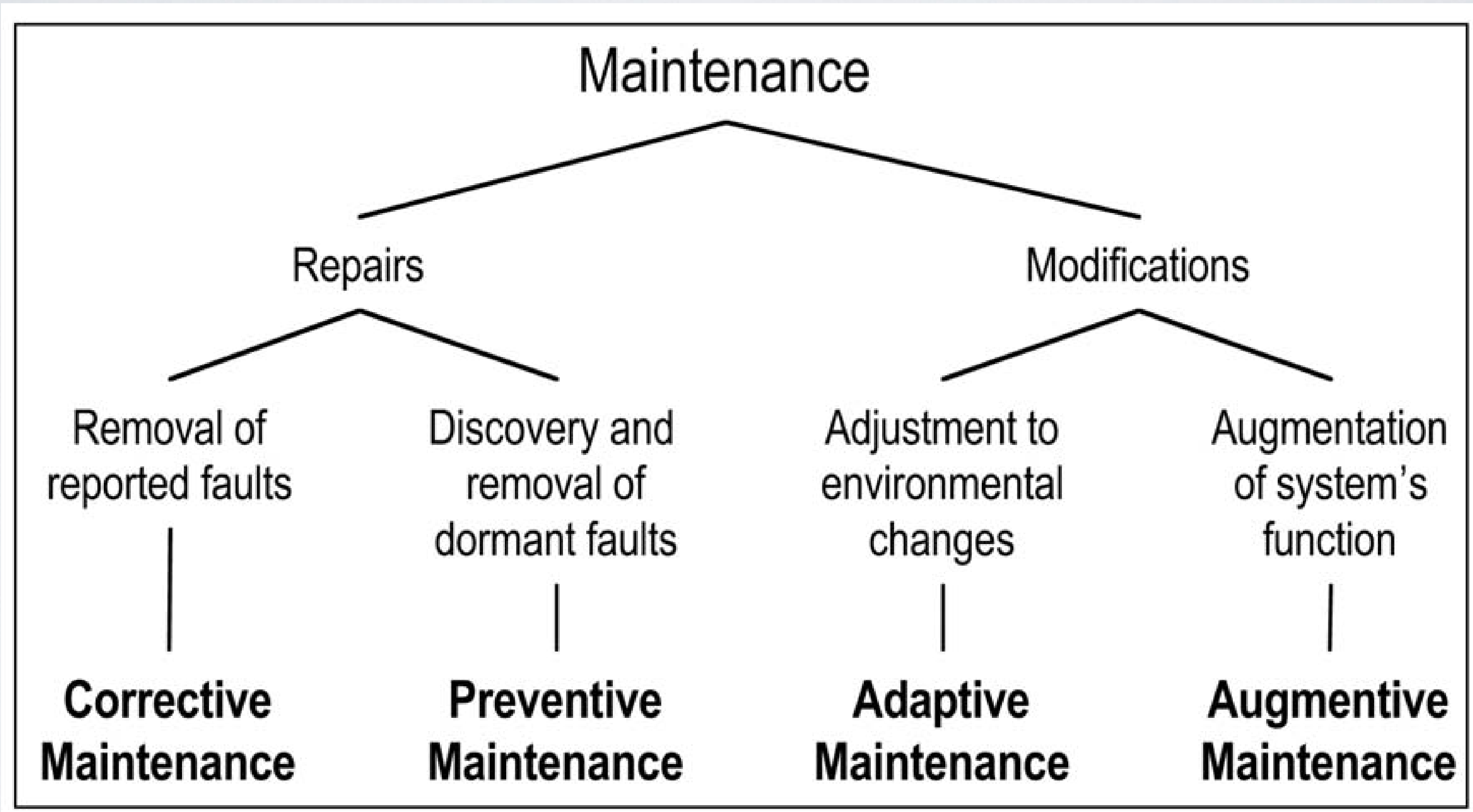
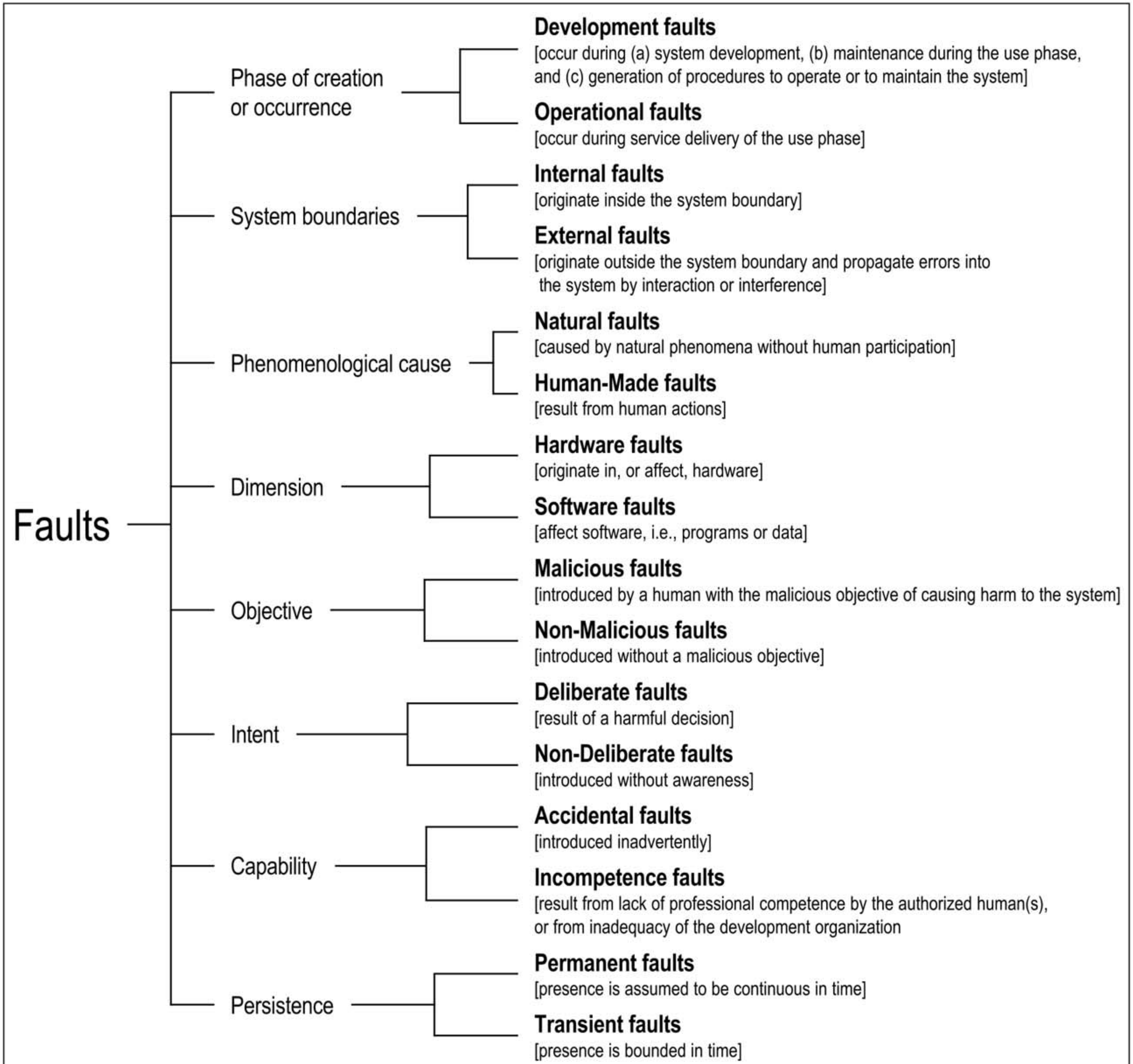
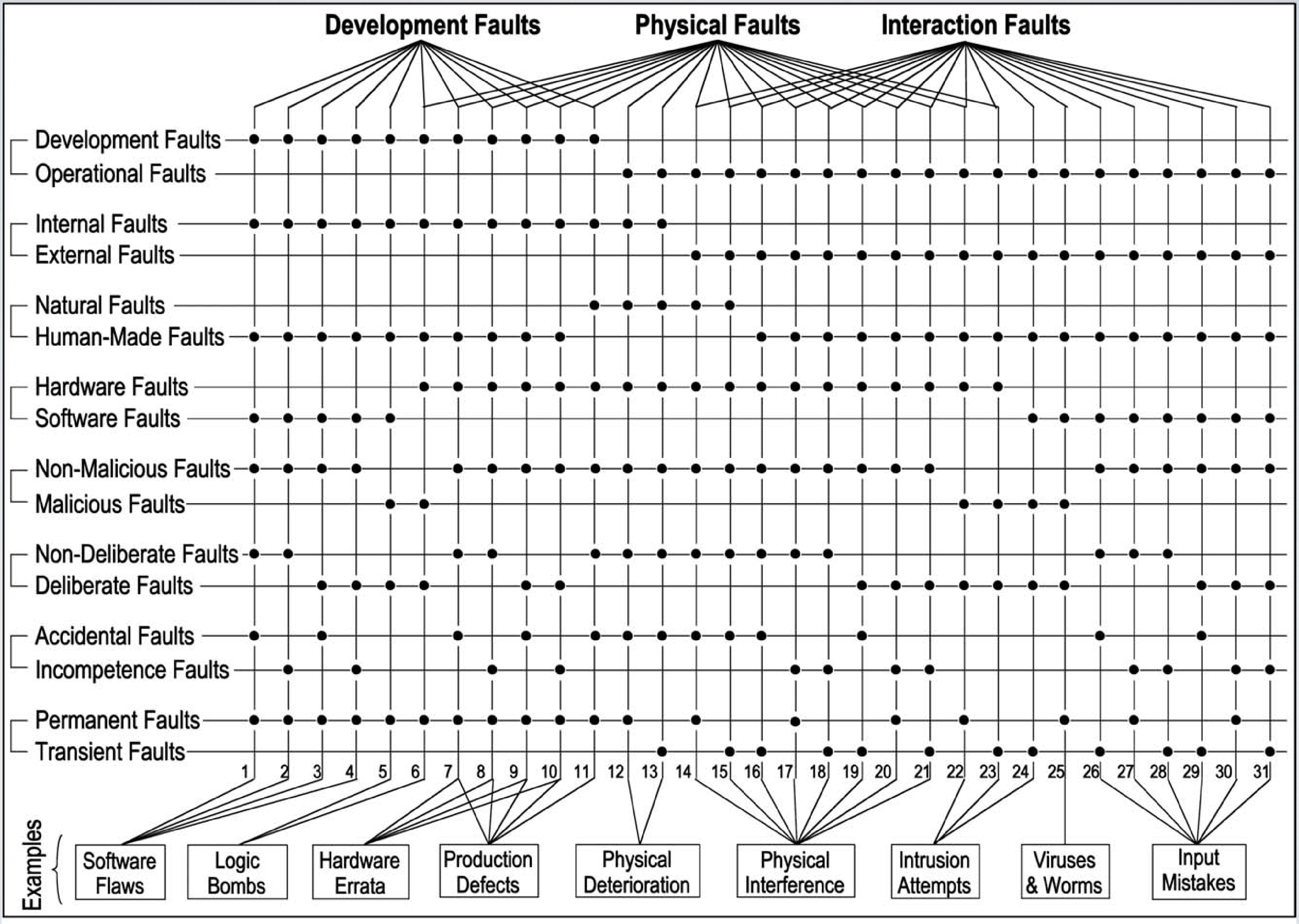


Fig. 3. The various forms of maintenance.

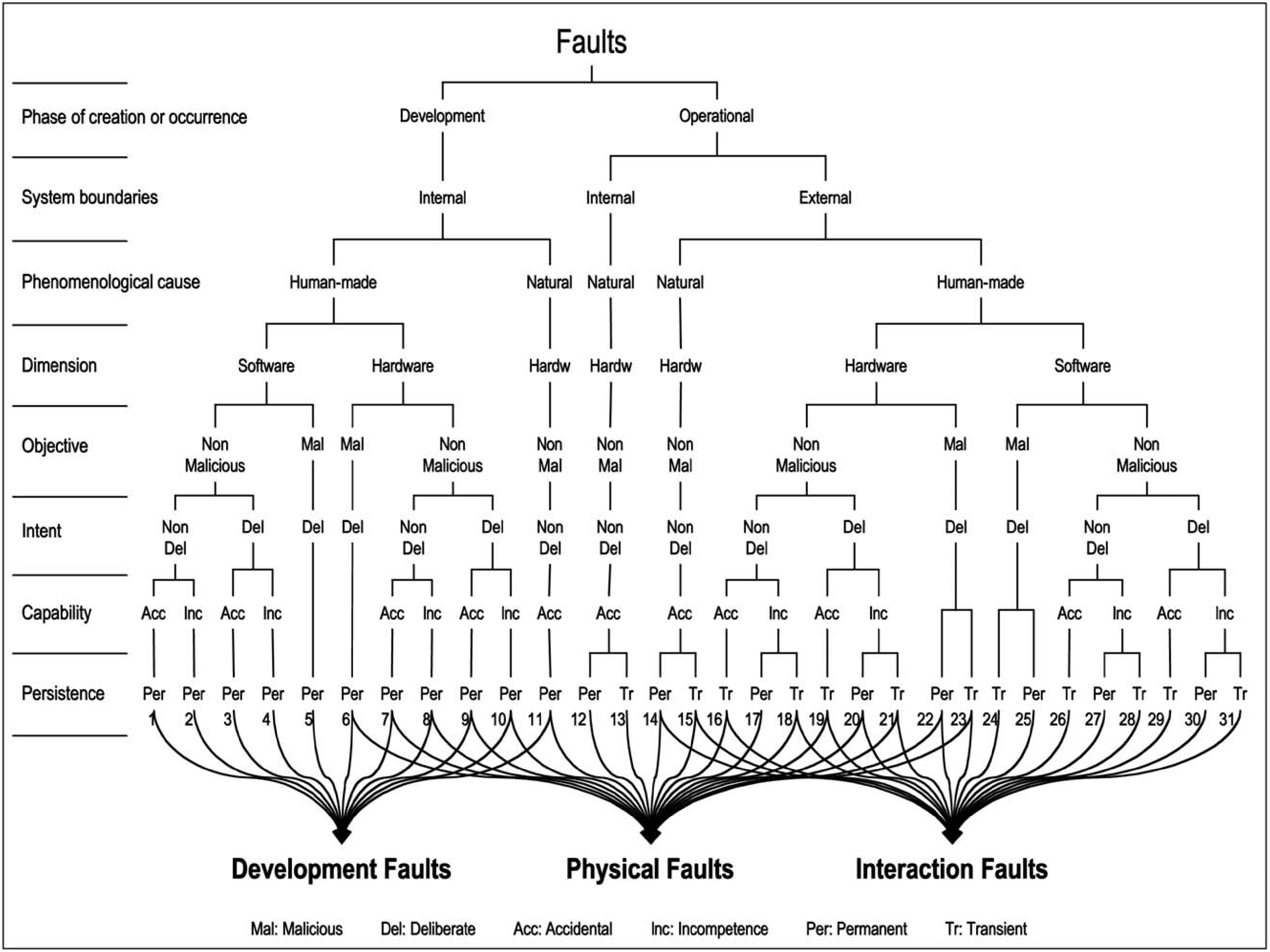
FAULTS: OVERVIEW

FIG:4 ELEMENTARY FAULT CLASSES





Development Faults
Physical Faults
Interaction Faults
Development Faults
Operational Faults
Internal Faults
External Faults
Natural Faults
Human-Made Faults
Hardware Faults
Software Faults
Non-Malicious Faults
Malicious Faults
Non-Deliberate Faults
Deliberate Faults
Accidental Faults
Incompetence Faults
Permanent Faults
Transient Faults
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Examples
Software Flaws
Logic Bombs
Hardware Errata
Production Defects
Physical Deterioration
Physical Interference
Intrusion Attempts
Viruses & Worms
Input Mistakes



Faults
Phase of creation or occurrence
System boundaries
Phenomenological cause
Dimension
Objective
Intent
Capability
Persistence
Development
Operational
Internal
Internal
External
Human-made
Natural
Natural
Natural
Human-made
Software
Hardware
Hardw
Hardw
Hardw
Hardware
Software
Non Malicious
Mal
Non Mal
Non Del
Del
Acc
Inc
Per
Tr
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Development Faults
Physical Faults
Interaction Faults
Mal: Malicious
Del: Deliberate
Acc: Accidental
Inc: Incompetence
Per: Permanent
Tr: Transient

3.2.3 ON HUMAN-MADE FAULTS

- Non-malicious faults
  - introduced without malicious objectives
  - **non-deliberate fault**: due to *mistakes*, i.e., *unintended action*, developer/operator/ maintainer is not aware
  - **deliberate fault**: due to *bad decisions*, i.e., *unintended actio*n that are wrong and cause faults

3.2.3 ON HUMAN-MADE FAULTS

- Non-malicious faults
  - further partitioning into:
  - **accidental faults**
  - **incompetence faults**

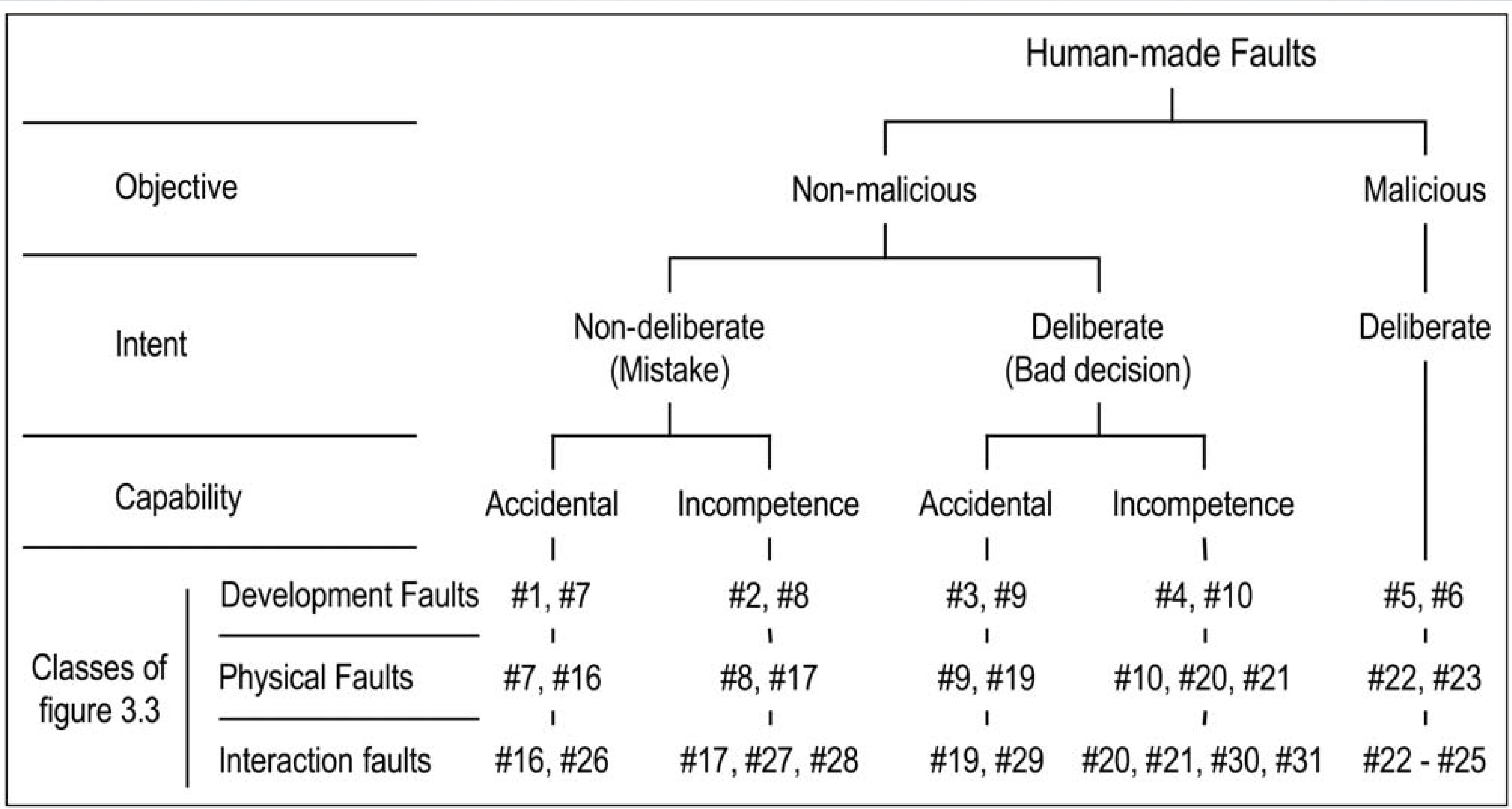


Fig. 6. Classification of human-made faults.

NON-MALICIOUS FAULTS

- Incompetence faults
  - individual, group, organization
  - e.g., Advance Automation System to replace aging USA air traffic control system

NON-MALICIOUS FAULTS

- Deployment faults
  - hardware
    - e.g., HW "errata" are listed in specification updates
    - may continue during lifetime of the product
  - software
    - software aging: progressively accrued error conditions cause performance degradation of failure
    - e.g., memory bloating/leaking, unterminated threads, storage space fragmentation, accumulation of round-off errors, ...

3.2.4 ON MALICIOUS FAULTS

- Malicious human-made faults
  - typical goals:
    - disrupt or halt service => denial of service
    - access confidential information
    - improperly modify the systems

3.2.4 ON MALICIOUS FAULTS

- Malicious logic faults
  - development faults: e.g., Trojan horses, logic or timing bombs, trapdoors
  - operational faults: e.g. viruses, worms, zombies
- Intrusion attempts
  - operational external faults. May be performed by system operators/admins
  - may use physical means to cause faults, e.g., power fluctuation, radiation, wire-tapping, heating/cooling

Fig 7.

Malicious

logic

faults

logic bomb: *malicious logic* that remains dormant in the host system till a certain time or an event occurs, or certain conditions are met, and then deletes files, slows down or crashes the host system, etc.

Trojan horse: *malicious logic* performing, or able to perform, an illegitimate action while giving the impression of being legitimate; the illegitimate action can be the disclosure or modification of information (attack against confidentiality or integrity) or a *logic bomb*;

trapdoor: *malicious logic* that provides a means of circumventing access control mechanisms;

virus: *malicious logic* that replicates itself and joins another program when it is executed, thereby turning into a *Trojan horse*; a virus can carry a *logic bomb*;

worm: *malicious logic* that replicates itself and propagates without the users being aware of it; a worm can also carry a *logic bomb*;

zombie: *malicious logic* that can be triggered by an attacker in order to mount a coordinated attack.

3.2.5 ON INTERACTION FAULTS

- Occur in use phase
  - elements of the use environment interaction with the system
  - all *external*
  - human-made
- Examples
  - configuration faults, reconfiguration faults

3.3 FAILURES

- Service failure
  - def.: event that occurs when the delivered service deviates from correct service
  - *service failure modes*: different ways in which deviation is manifested
  - *content failure*: content of info delivered deviates from implementing the system function
  - *timing failure:* time of arrival (early or late) or duration of info delivered at service interface deviates from implementing the system function.

3.3 FAILURES

- Service failure cont.
  - both information and timing are incorrect:
    - *halt failure:* external state becomes constant
    - *silent failure:* no service is delivered at interface
    - *erratic failure:* service is delivered (not halted) but is erratic, e.g. babbling

3.3 FAILURES

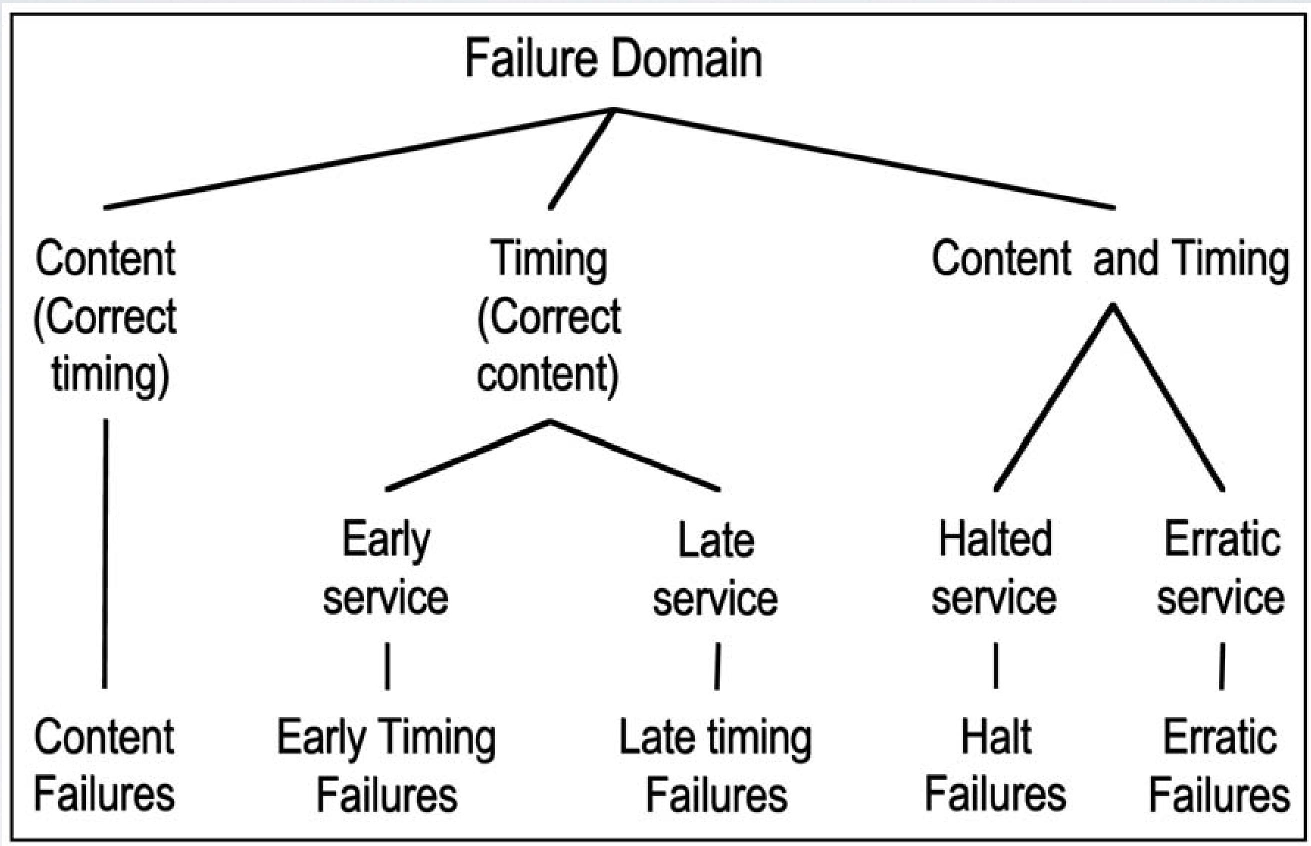


Fig. 8. Service failure modes with respect to the failure domain viewpoint.

3.3 FAILURES

- Consistency
  - consistent failures: incorrect service is perceived identically by all system users
  - inconsistent failures: some of all users perceive differently incorrect service. Byzantine failures

SERVICE FAILURE MODES

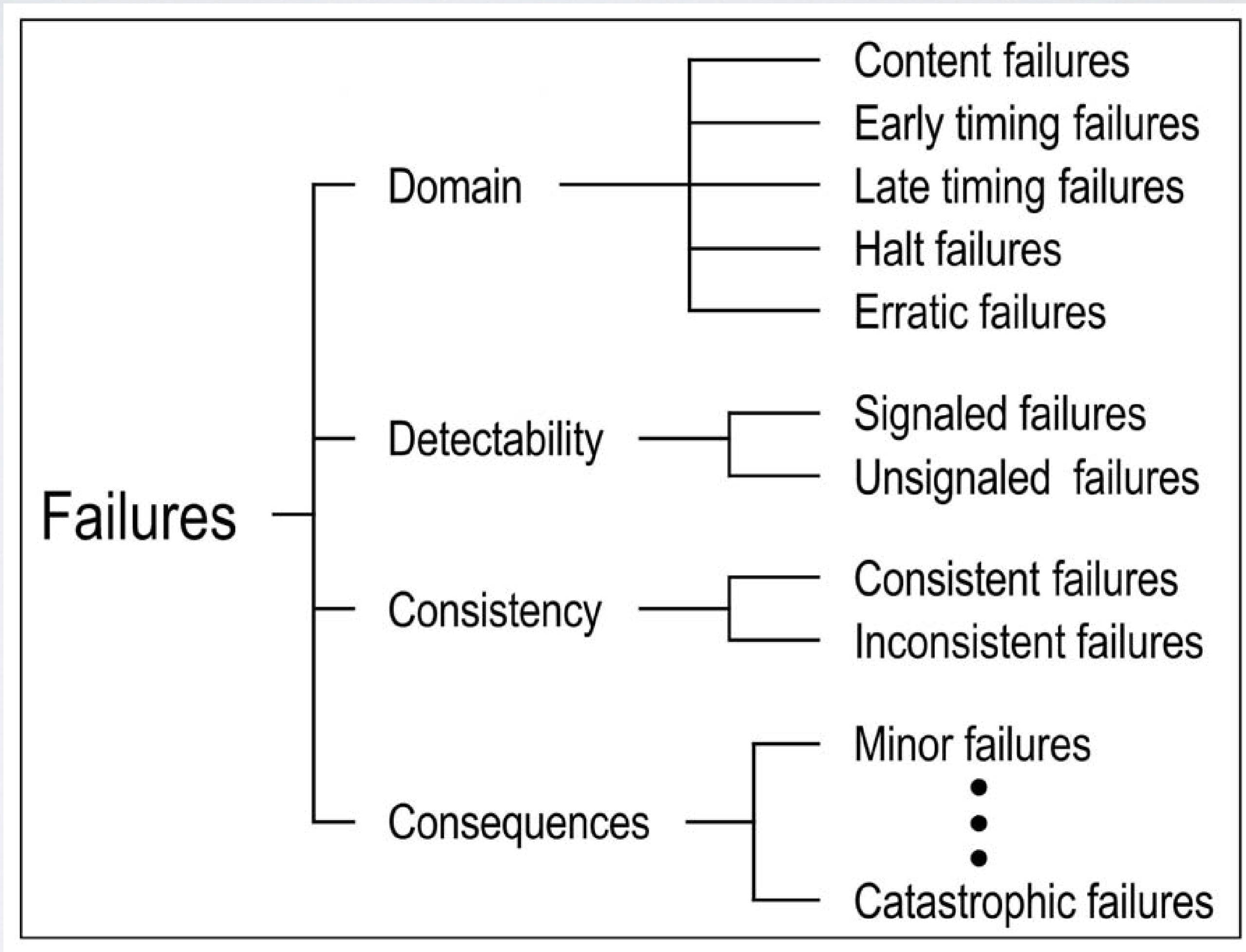


Fig. 9. Service failure modes.

3.3.2 DEVELOPMENT FAILURES

- Budget failure
  - "broke" before system passes acceptance testing
- Schedule failure
  - schedule slips to a point in the future where the system would be technologically obsolete or functionally inadequate for user's needs

3.5 FAULTS, ERRORS AND FAILURES

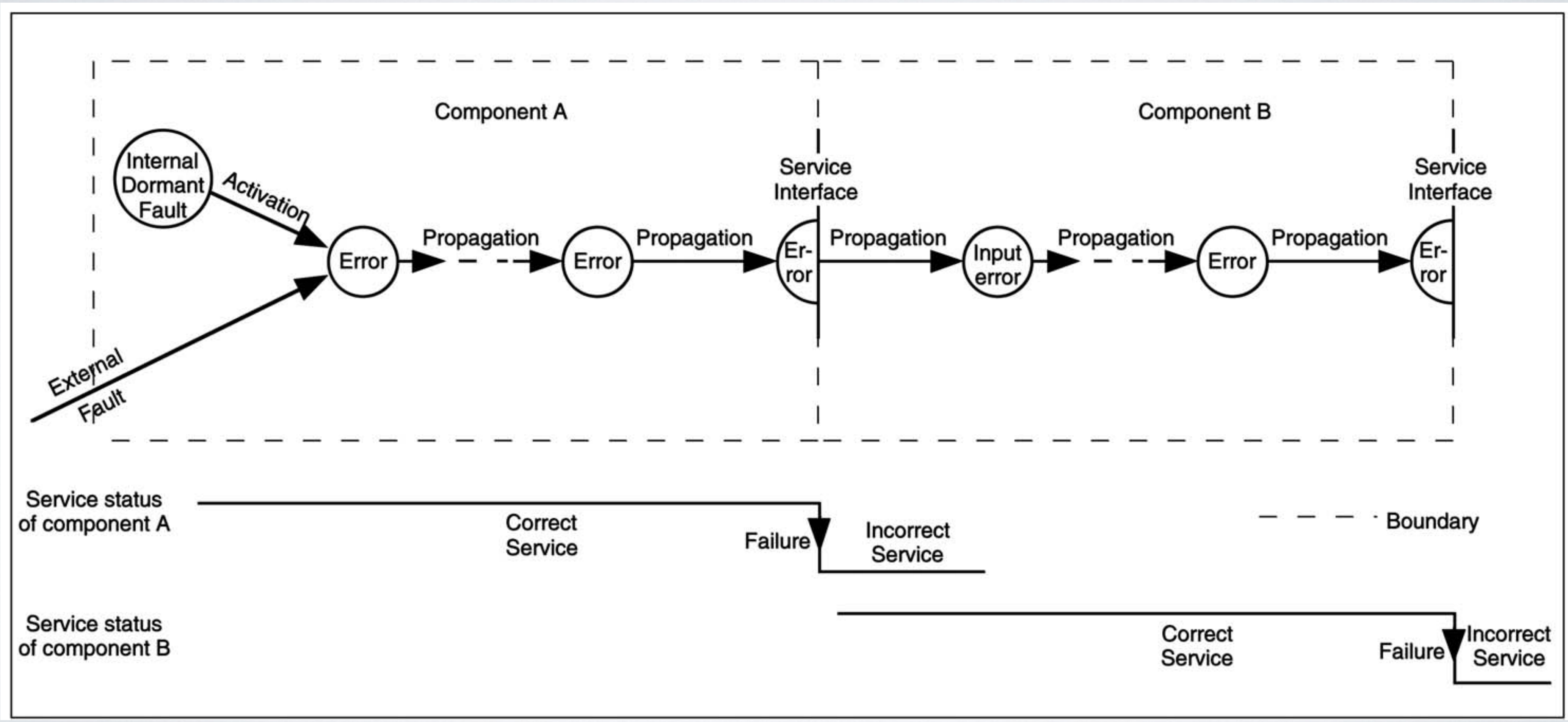


Fig. 10 Error Propagation

EXAMPLES

- traditional hardware fault tolerance view
  - physical fault (may be dormant), e.g., stuck-at
  - produces error
  - may result in failure

EXAMPLES

- programming "bug"
    - *error* by programmer leads to *failure* to write the correct instruction or data
    - this results in a (*dormant*) *fault* in code or data
    - upon activation the *fault* becomes *active* and produces an *error*
    - this *error* may result in *failure*

EXAMPLES

- Specification related
  - *error* by a specifier leads to *failure* to describe a function
  - this results in a *fault* in a written specification, e.g., incomplete description of a function.
  - this incomplete function may deliver service different from expected service
  - user perceives this as *error* resulting in *failure*

EXAMPLES

- Inappropriate human-system interaction
  - inappropriate human-system interaction performed by operator during operation of system
  - results in external *fault* (from system's viewpoint)
  - resulting altered processed data is an *error*...

EXAMPLES

- Reasoning
  - *error* in reasoning leads to a maintenance or operating manual writer's *failure* to write correct directives
  - results in a *fault* in the manual (faulty directives) that will remain *dormant* as long as the directives are not acted upon...

EXAMPLES

- Combined action of several faults
  - consider trap-door (by-pass access control)
  - this is a development *fault*
  - remains *dormant* until exploited
  - intruder login is deliberate interaction *fault*
  - intruder may create an *error* -> service affected -> *failure*

HARD AND SOFT FAULTS

- Hard (or solid) faults
  - fault activation is reproducible
- Soft (or elusive) faults
  - not systematically reproducible

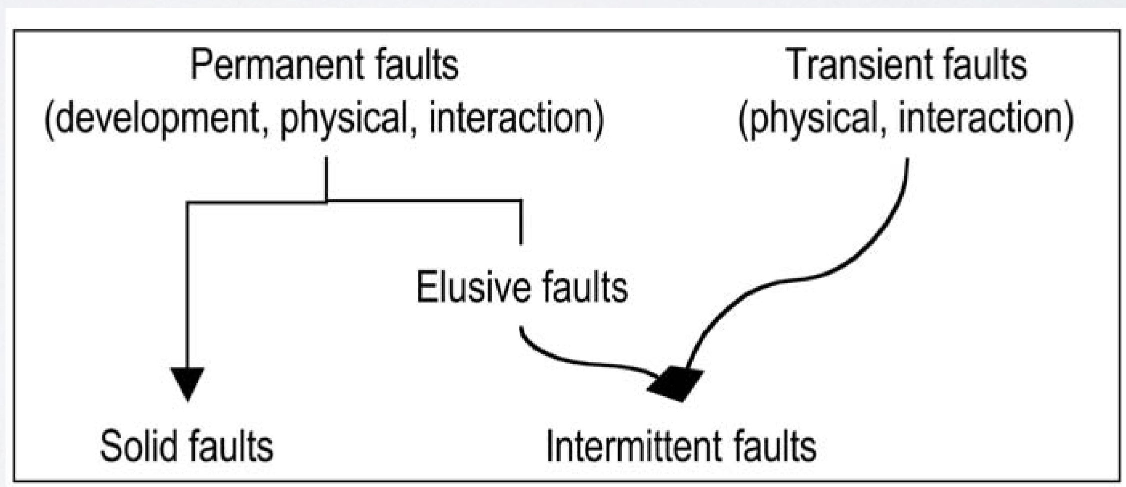


Fig. 13. Solid versus intermittent faults.

4. DEPENDABILITY AND SECURITY

- From definition point of view

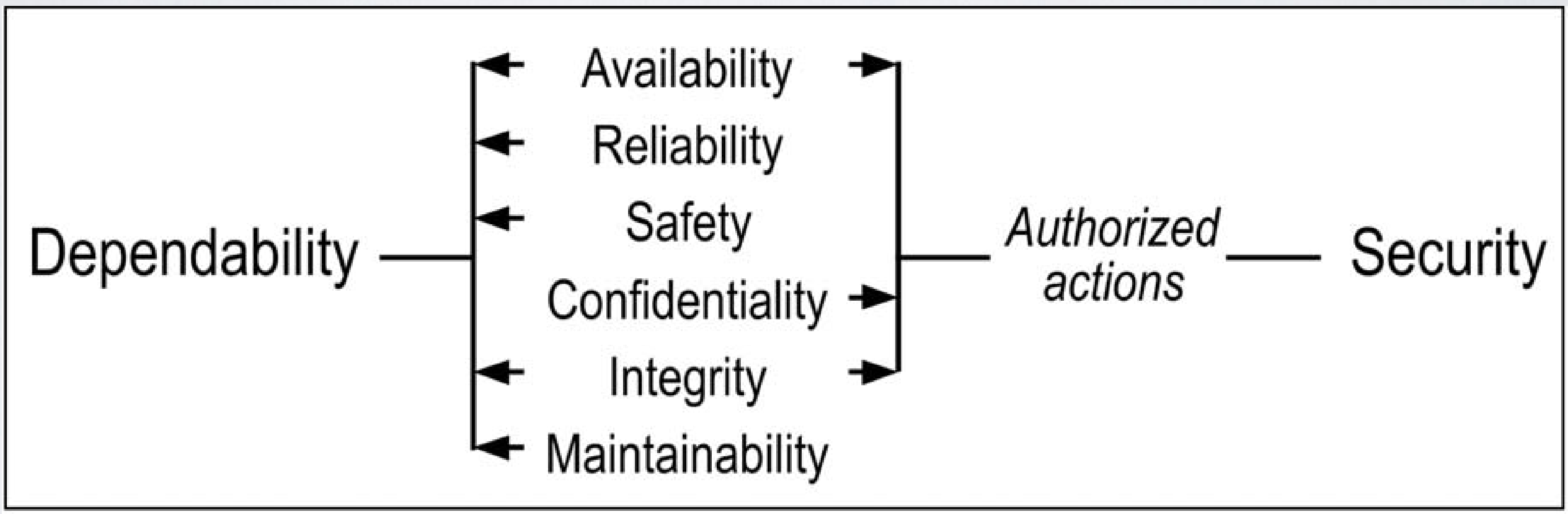


Fig. 14. Relationship between dependability and security.

4. DEPENDENCE AND TRUST

- Dependence
  - The dependence of system A on system B represents the extent to which System A's dependability is (or would be) affected by that of System B.
  - a component *a* depends upon a component *b* if the correctness of *b*'s service delivery is necessary for the correctness of *a*'s service delivery.
- Trust
  - Trust is accepted dependence.

4. DEPENDENCE AND TRUST

- Levels of dependence
  - from total dependence to complete independence
- Accepted dependence
  - judgement that level of dependence is acceptable
  - judgement possibly explicit, e.g., contract between "parties"
  - judgement may be unwilling, e.g., there is no other option!
  - the extent to which A fails to provide means of tolerating B's failures is a measure of A's (perhaps unthinking or unwilling) trust in B.

4.3 ATTRIBUTES OF DEP. & SEC.

- Availability, integrity, maintainability, reliability, safety, confidentiality...
  - Don't think binary, absolute, or deterministic
  - Do think relative and probabilistic

Fig. 15. Dependability, high confidence, survivability, and trustworthiness.

Concept	Dependability	High Confidence	Survivability	Trustworthiness
Goal	1) ability to deliver service that can justifiably be trusted 2) ability of a system to avoid service failures that are more frequent or more severe than is acceptable	consequences of the system behavior are well understood and predictable	capability of a system to fulfill its mission in a timely manner	assurance that a system will perform as expected
Threats present	1) development faults (e.g., software flaws, hardware errata, malicious logic) 2) physical faults (e.g., production defects, physical deterioration) 3) interaction faults (e.g., physical interference, input mistakes, attacks, including viruses, worms, intrusions)	• internal and external threats • naturally occurring hazards and malicious attacks from a sophisticated and well-funded adversary	1) attacks (e.g., intrusions, probes, denials of service) 2) failures (internally generated events due to, e.g., software design errors, hardware degradation, human errors, corrupted data) 3) accidents (externally generated events such as natural disasters)	1) hostile attacks (from hackers or insiders) 2) environmental disruptions (accidental disruptions, either man-made or natural) 3) human and operator errors (e.g., software flaws, mistakes by human operators)
Reference	This paper	"Information Technology Frontiers for a New Millennium (Blue Book 2000)" [48]	"Survivable network systems" [16]	"Trust in cyberspace" [62]

5.1 FAULT PREVENTION

- General engineering
  - e.g., prevention of development faults
  - development methodologies
    - SW: e.g., information hiding, modularization strongly-typed programming languages
    - HW: e.g., design rules

5.1 FAULT TOLERANCE

- Concepts
  - Diagnosis
  - Rollback recovery
  - Forward recovery
  - Fault masking
  - How are these concepts related?

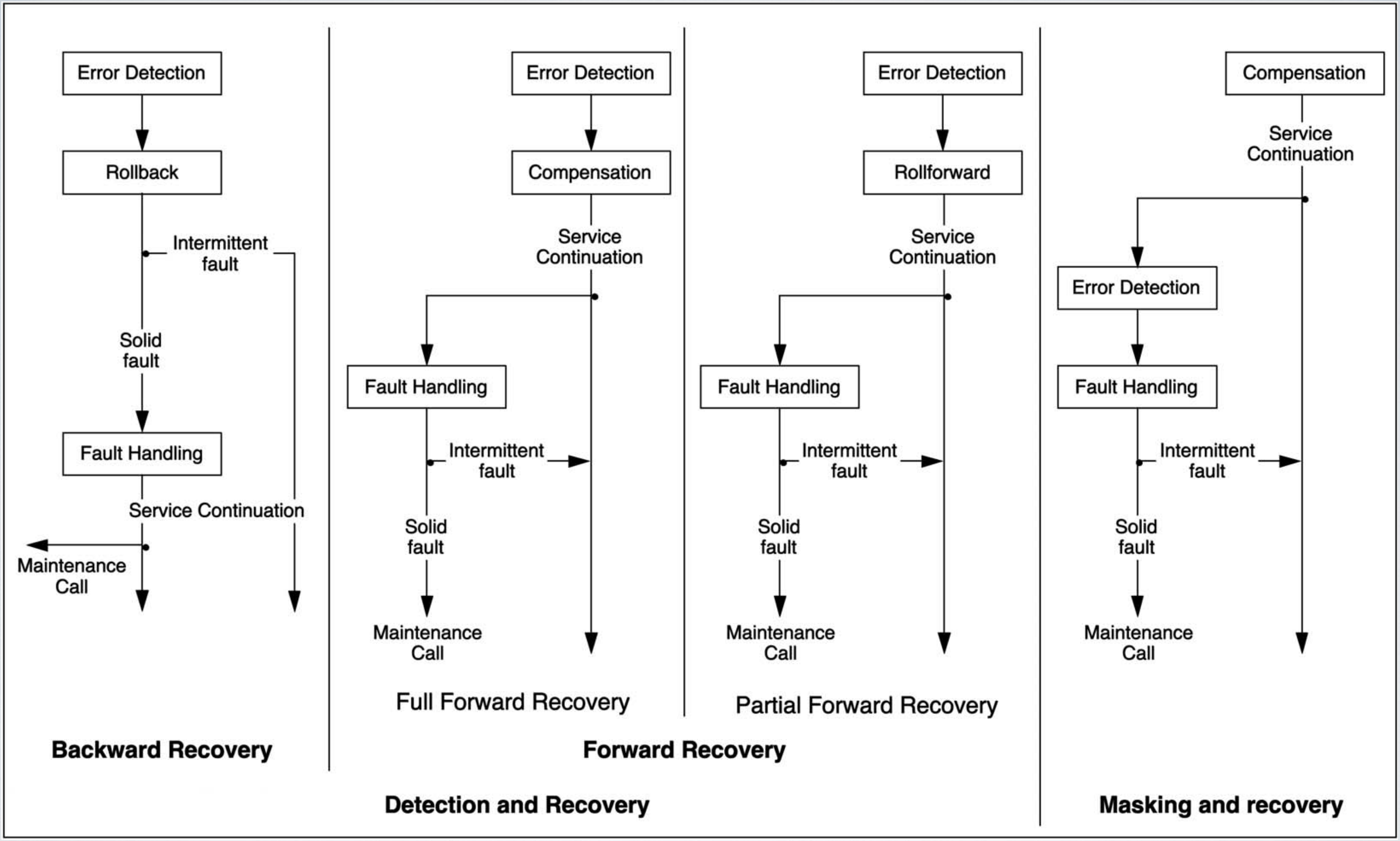
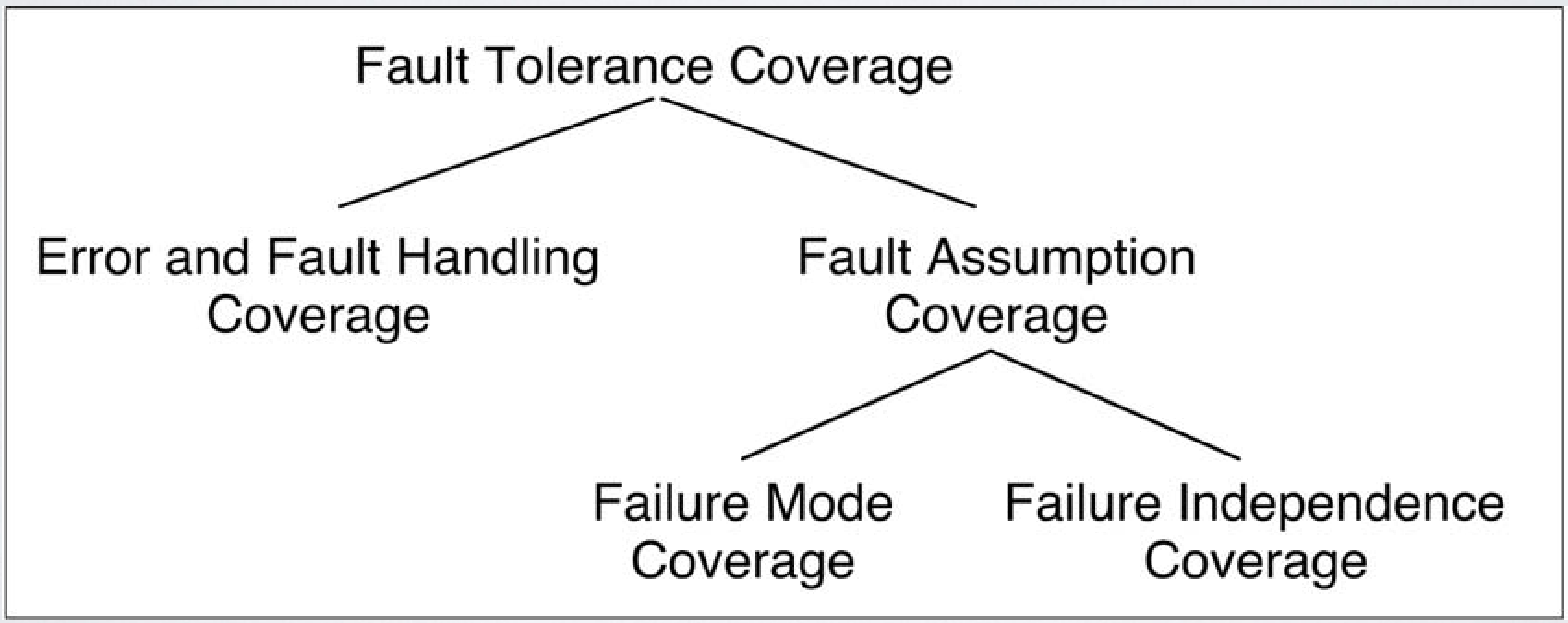


Fig. 17. Examples for the basic strategies for implementing fault tolerance.

FAULT COVERAGE

• Fi



5.3 FAULT REMOVAL

- During Development
  - Verification
    - the process of checking whether the system adheres to given properties, termed the verification conditions
  - Diagnosis
    - diagnosing the fault(s) that prevented the verification conditions from being fulfilled
  - Correction
    - after correction repeat verification: nonregression verification

5.3 FAULT REMOVAL

- Static Verification
  - Verification without actual execution
  - On System:
    - use static analysis
    - theorem proving
  - On Model of system behavior
    - model checking: state transition model
    - e.g., Petri net, state automata

SIDE NOTE

- What is the relationship between Specification and what has been implemented?
  - discussion on mapping in two directions

VERIFICATION APPROACHES

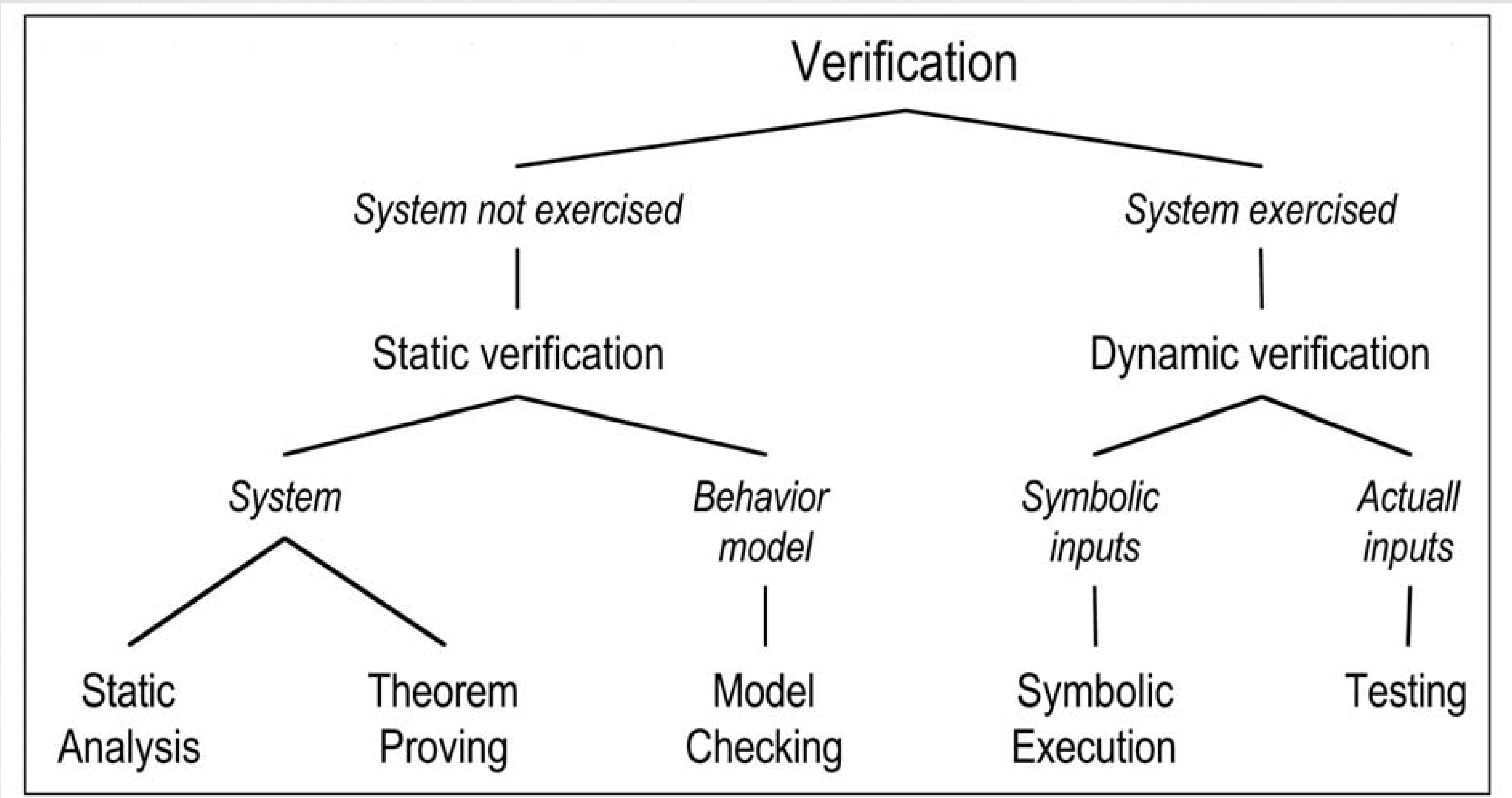
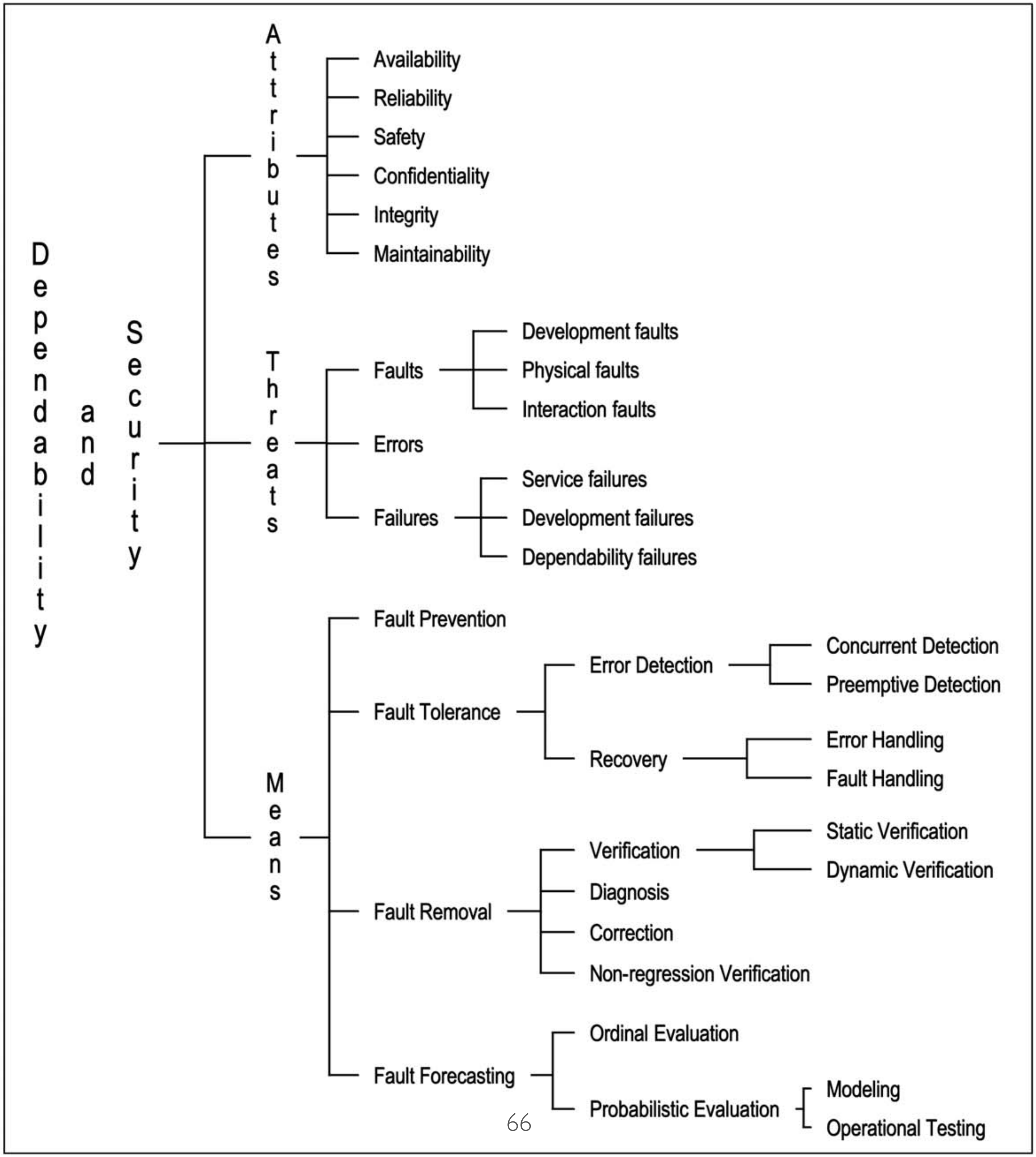


Fig. 19. Verification approaches.

5.4 FAULT FORECASTING

- Predictive approach
  - **qualitative evaluation**, aims to identify, classify, and rank the failure modes, or the event combinations (component failures or environ- mental conditions) that would lead to system failures;
  - **quantitative (or probabilistic) evaluation**, aims to evaluate in terms of probabilities the extent to which some of the attributes are satisfied; those attributes are then viewed as measures.


Dependability and Security
Attributes
Availability
Reliability
Safety
Confidentiality
Integrity
Maintainability
Threats
Faults
Development faults
Physical faults
Interaction faults
Errors
Failures
Service failures
Development failures
Dependability failures
Means
Fault Prevention
Fault Tolerance
Error Detection
Concurrent Detection
Preemptive Detection
Recovery
Error Handling
Fault Handling
Fault Removal
Verification
Static Verification
Dynamic Verification
Diagnosis
Correction
Non-regression Verification
Fault Forecasting
Ordinal Evaluation
Probabilistic Evaluation
Modeling
Operational Testing