



UNIVERSITÀ DI PISA

Dependable and secure computing

Basic concepts and terminology

(in this part of the course we give precise definitions of the concepts that come into play when addressing the dependability of computing systems)

A. Avizienis, J.C. Laprie, B. Randell, C. Landwehr

Basic Concepts and Taxonomy of Dependable and Secure Computing

IEEE Transactions on Dependable and Secure Computing, Vol. 1, N. 1, 2004

- Dependability
- The dependability tree
- Chain of threats: faults, errors, failures
- Classification of faults
- Errors
- Classification of failures

Dependability: a definition



A system is designed to provide a certain service

Dependability is the ability of a system to deliver the correct service also in presence of faults and malfunctions

Dependability is “that property of a computer system such that reliance can justifiably be placed on the service it delivers”

(this definition stresses the need for justification of trust)

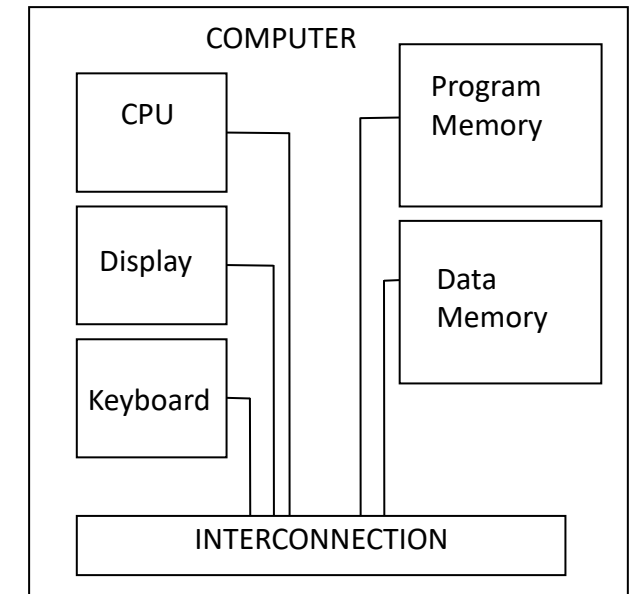
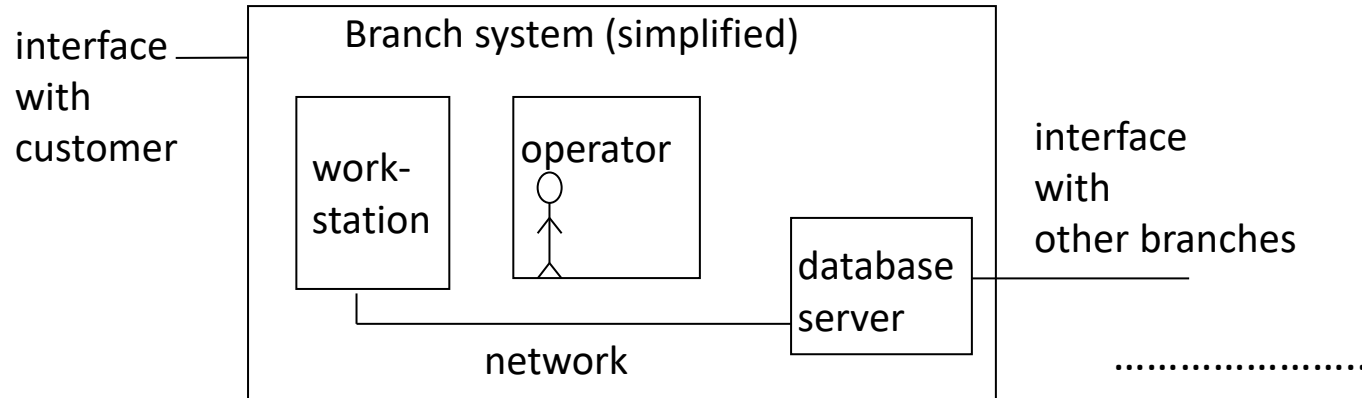
Systems and components



A system is made out of components. Each component is a system in its own right.

Bank information system

Providing services to customers and to other branches



Computer-based systems failures



If the system stops delivering the intended service, we call this a failure



ATM (Automated Teller Machine) system

dispense the amount of cash requested by the customer

For instance:

deliver 200 Euro when the customer asked 20 Euro

The causes of failure are named faults

A fault causes an error in the state of the system

The error causes the system to fail

Failures may have many different causes (faults)

- chip suffers permanent electrical damage
- undersized fan (design fault) allows overheating on hot days
- Chip malfunction (physical fault)
- The machine works ok after cooling down (the fault is transient)
- Operator pushes the wrong button
- Cosmic ray particle causing transient upset in execution (bit flip in a memory)
- Defect in software

.....

Computer failures differ from failures of other equipments

- Subtler failures than “breaking down” or “stopping working”, ..
- The computer is used to store information:
 - there are many ways information can be wrong, many different effects both within and outside the computer
- Computing systems are complex hierarchies relaying on hidden components
- Small hidden faults may have large effects (digital machine)

System: entity that interacts with other sub-systems

- hardware
- networks
- operating systems software
- application software
- humans
- the physical world with its natural phenomena

Moreover, a failure can be the result of a system vulnerability and security attacks (malicious faults)

System: a precise definition



System is an entity of interest

System: A system is an entity that interacts with other entities, i.e., other systems, including hardware, software, humans and physical world with its natural phenomena.

The other systems are the environment of the given system.

System boundary: the system boundary is the common frontier between the system and its environment

System: a precise definition



System Requirements: define the problem that the computer system has to solve

- Functional requirements
- Dependability requirements

System function: what the system is intended to do.

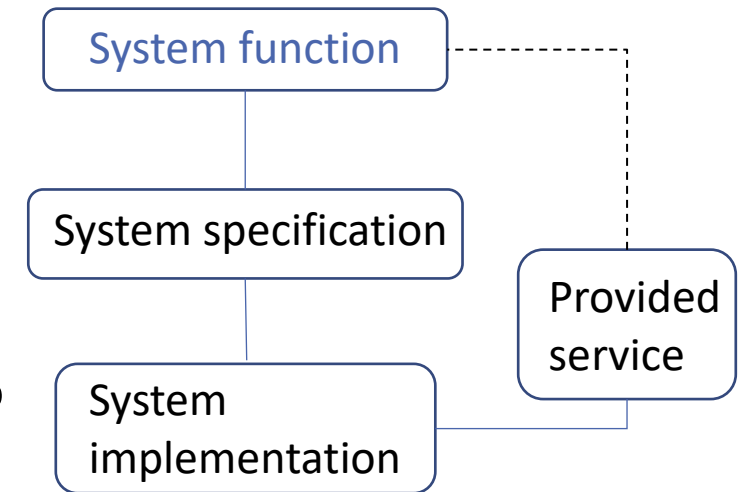
System Specification: define the solution that the computer system has to implement to solve the problem

<System function is described by the system specification>

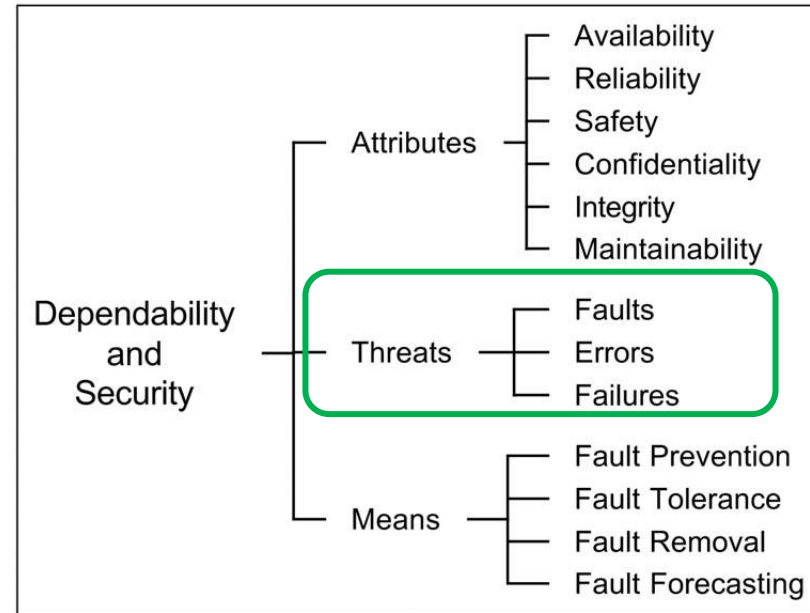
Correct service:

correct service is delivered when the service implements the **system function**

A service fails if either it does not comply with the system specification, or because the specification did not adequately describe the **system function**.



Dependability tree



From [Avizienis et al., 2004]

(*) Security: Availability, Confidentiality, Integrity

Threats to Dependability: precise definition



Fault

the adjudged or hypothesized cause of problems

Error

a fault first causes an error in the service state of a component that is a part of the internal state of the system

Failure

the external state is affected by the error

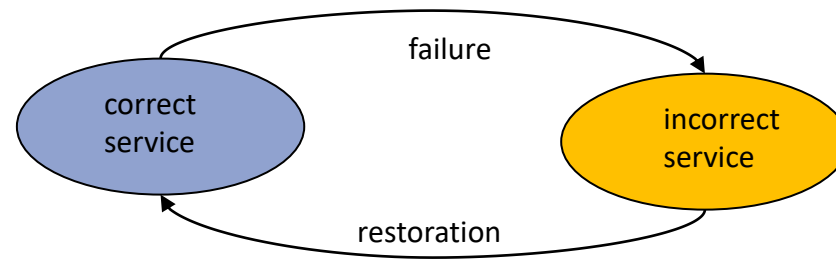
For this reason, the definition of an error is the part of the total state of the system that may lead to its subsequent service failure

It is important to note that many errors do not reach the system's external state and cause a failure

Threats to Dependability



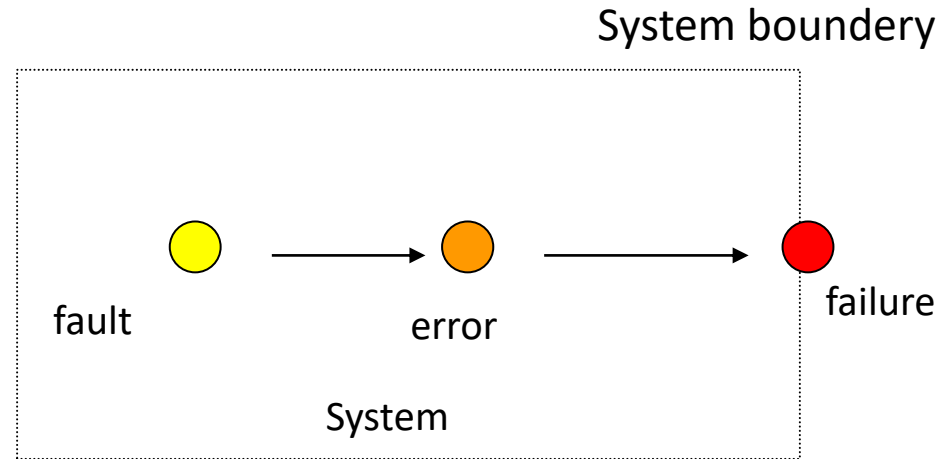
- **Correct service** is delivered when the service implements the system function
- A **service failure**, often abbreviated failure, is an event that occurs when the delivered service deviates from correct service
- Failure is a transition from correct service to incorrect service
- Restoration is the transition from incorrect service to correct service.



Threats to Dependability



UNIVERSITÀ DI PISA



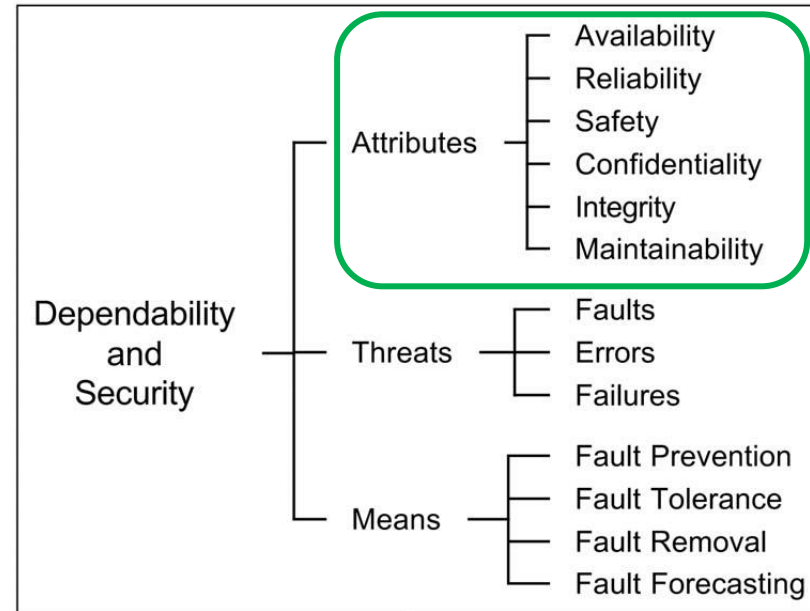
A fault causes an error in the internal state of the system.
The error causes the system to fail

It is important to note that many errors do not reach the system's external state and cause a failure.

Dependability attributes



Dependability is a concept that encompasses multiple properties (attributes)



From [Avizienis et al., 2004]

(*) Security: Availability, Confidentiality, Integrity

Dependability attributes



UNIVERSITÀ DI PISA

- **Availability**

readiness for correct service

- **Reliability**

continuity of correct service

- **Safety**

absence of catastrophic consequences on the user(s) and the environment

- **Confidentiality**

the absence of unauthorized disclosure of information

- **Integrity**

absence of improper system alterations

- **Maintainability**

ability to undergo modifications and repairs

Dependability attributes
can be measured in terms
of probability

Dependability attributes



Alternative definition of dependability that provides the criterion for deciding if the system is dependable or not

The dependability of a system is the ability to avoid service failures that are more frequent and more severe than is acceptable

The system requirements must include the requirements for the dependability attributes in terms of the acceptable frequency and severity of service failures for specified classes of faults and a given use environment.

One or more attributes may not be required at all for a given system

Threats to Dependability: system life cycle



Life cycle of a system

- development phase
- use phase

Development phase includes all activities from presentation of the user's initial concept to the decision that the system has passed all acceptance tests and is ready to deliver service in its user's environment.

The **use phase** of a system's life begins when the system is accepted for use and starts the delivery of its services to the users.

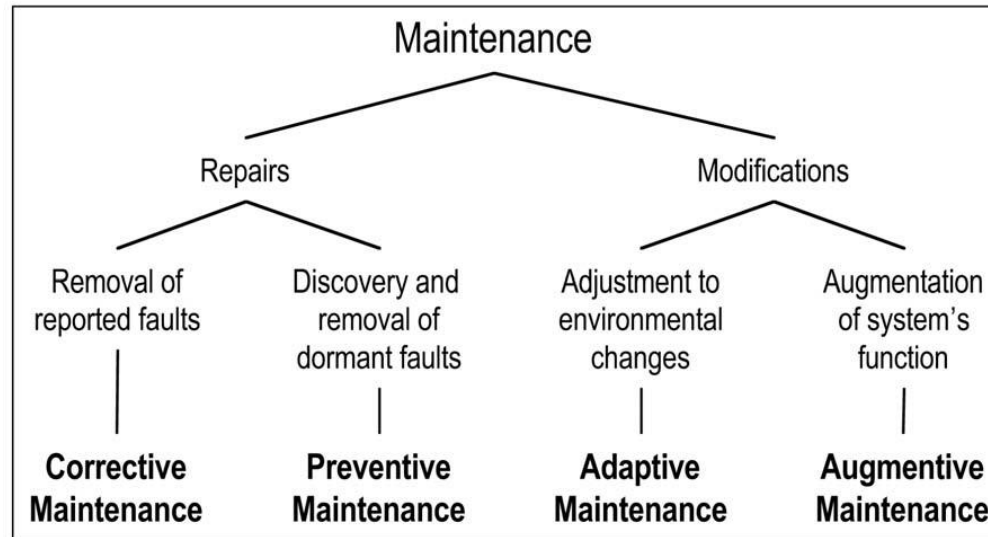
Threats to Dependability: system life cycle

- The **use phase** consists of alternating periods of **correct service** delivery (to be called service delivery), **service outage**, and service **shutdown**.
- A **service outage** is caused by a service failure. It is the period when incorrect service (including no service at all) is delivered at the service interface.
- A **service shutdown** is an intentional halt of service by an authorized entity.
- **Maintenance** actions may take place during all three periods of the use phase. Maintenance includes not only **repairs**, but also all **modifications** of the system that take place during the use phase of system life.

Threats to Dependability: Maintenance



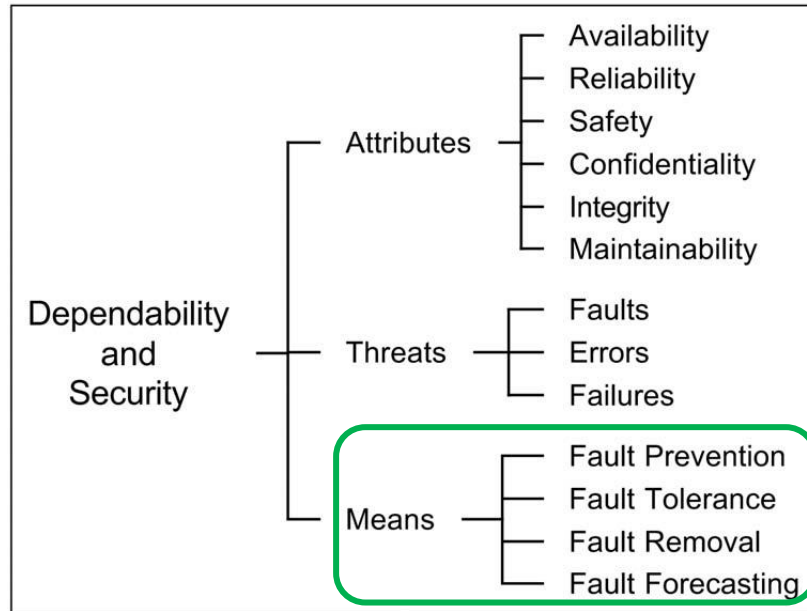
UNIVERSITÀ DI PISA



From [Avizienis et al., 2004]

Maintenance involves the participation of an external agent
e.g., a repairman, test equipment, remote reloading of software

Dependability means



From [Avizienis et al., 2004]

Dependability means: approaches to dealing with faults

Means for achieving dependability



A combined use of methods can be applied as means for achieving dependability. These means can be classified into four categories:

1. Fault Prevention

- techniques to prevent the occurrence and introduction of faults
 - rigorous development, formal methods, testing,...
 - component screening, shielding, ...

2. Fault Tolerance techniques

deal with faults at run-time, deliver correct service in presence of activated faults and errors

3. Fault Removal techniques

techniques to remove faults in such a way that they are no more activated

4. Fault Forecasting techniques

to estimate the present number, the future incidence, and the consequences of faults



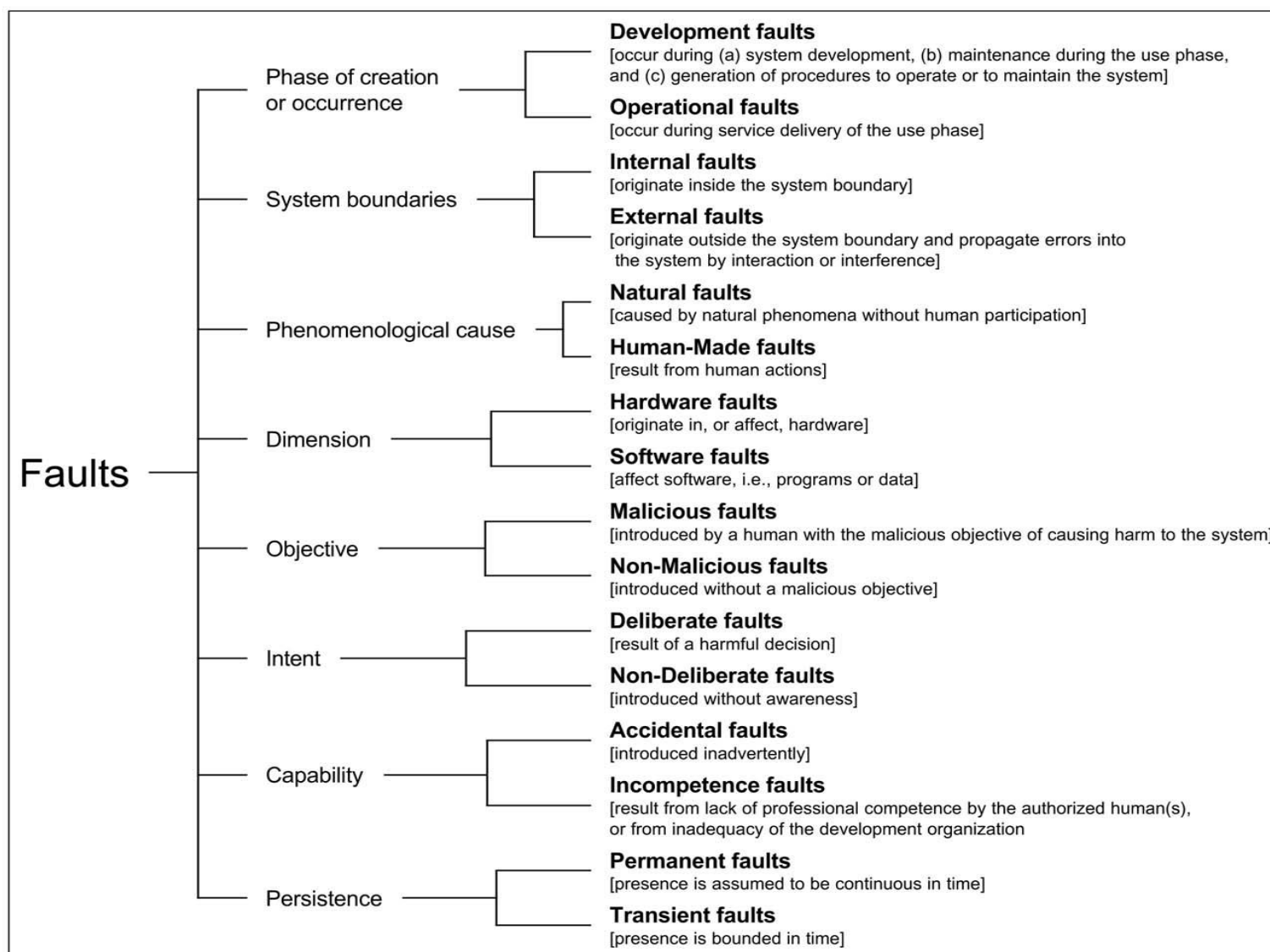
A taxonomy of faults

Faults classification



- All different faults that may affect a system during its life cannot be enumerated
- We can classify faults. Classification of faults is important because we can identify which mechanisms protect us from a given class of faults.
- Faults are classified according to basic viewpoints

Faults classification



From [Avizienis et al., 2004]

Identified combinations

three major partially overlapping groupings

- **Development faults**

that include all fault classes occurring during development

- **Physical faults**

that include all fault classes that affect hardware

- **Interaction faults**

that include all external faults

(31 combinations have been identified)

Identified combinations

three major partially overlapping groupings

- **Development faults**

that include all fault classes occurring during development

- **Physical faults**

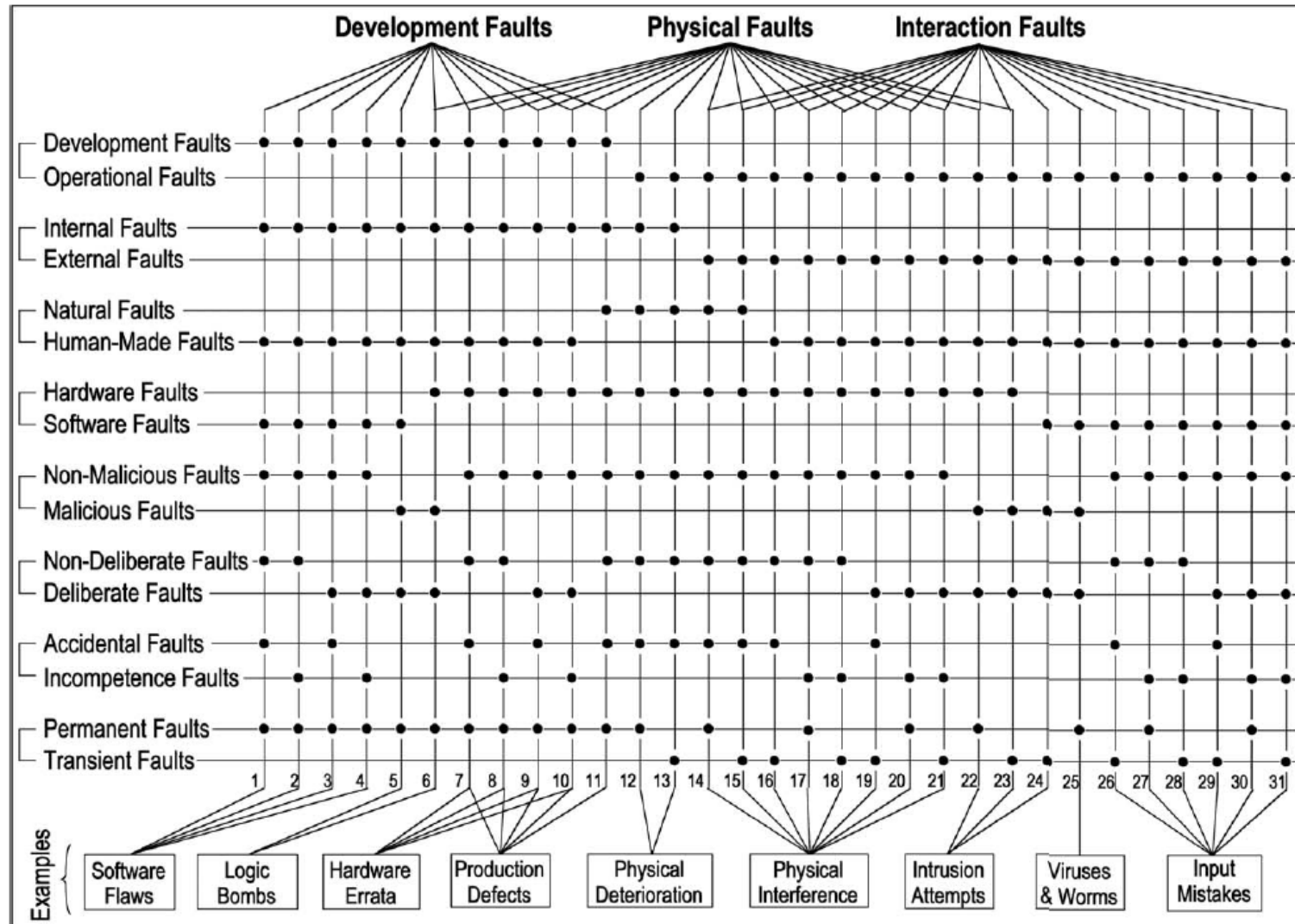
that include all fault classes that affect hardware

- **Interaction faults**

that include all external faults

(31 combinations have been identified)

Faults classification

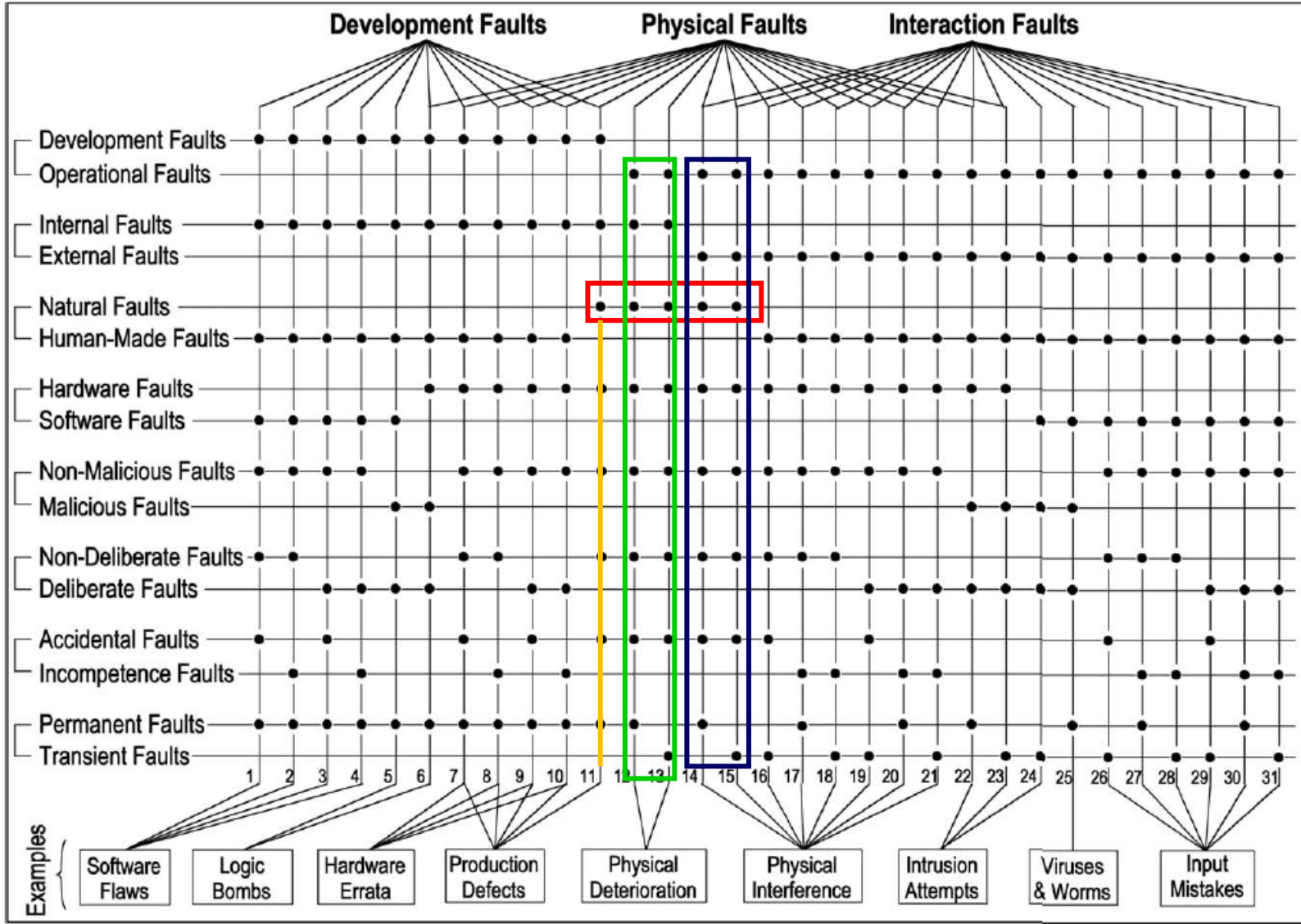


names of some illustrative fault classes

From [Avizienis et al., 2004]

- Natural faults (11-15) are physical (hardware) faults that are caused by natural phenomena without human participation
- Production defects (11) are natural faults that originate during development.
- Natural faults during **operation** are
 - **internal** (12-13), due to natural processes that cause physical deterioration, or
 - **external** (14-15), due to natural processes that originate outside the system boundaries and cause physical interference
 - by penetrating the hardware boundary of the system (radiation, etc.) or
 - by entering via use interfaces (power transients, noisy input lines, etc.)

Natural faults

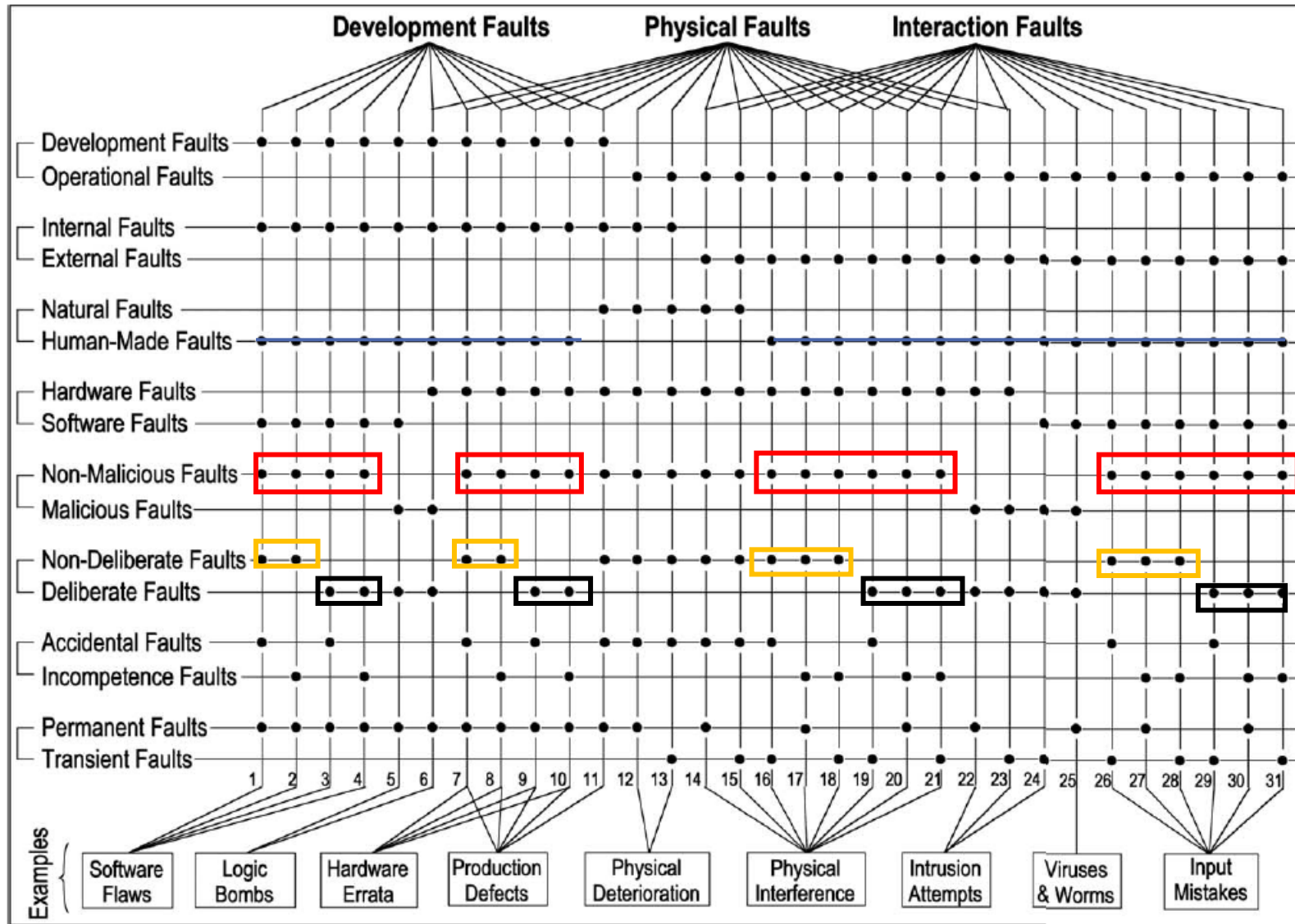


The two basic classes of human-made faults (**that result from human actions**) are:

Malicious faults, introduced during either system development with the objective to cause harm to the system during its use (5-6), or directly during use (22-25).

Nonmalicious faults (1-4, 7-10, 16-21, 26-31), introduced without malicious objectives.

Human-made
Non-malicious
Faults



- Non-malicious development faults are Software and Hardware faults.
- Hardware faults: microprocessor faults discovered after production (named Errata).
They are listed in specification updates

Non-malicious faults are:

1. **nondeliberate** faults that are due to mistakes, that is, unintended actions of which the developer, operator, maintainer, etc. is not aware (1, 2, 7, 8, 16-18, 26-28);
2. **deliberate** faults that are due to bad decisions, that is intended actions that are wrong and cause faults (3, 4, 9, 10, 19-21, 29-31)



Deliberate development faults (3, 4, 9, 10) result generally from trade offs, either 1) aimed at preserving acceptable performance, at facilitating system utilization, or 2) induced by economic considerations.

Deliberate interaction faults (19-21, 29-31) may result from the action of an operator either aimed at overcoming an unforeseen situation, or deliberately violating an operating procedure without having realized the possibly damaging consequences of this action

Deliberate faults

- are often recognized as faults only after an unacceptable system behavior; thus, a failure has ensued.
- the developer(s) or operator(s) did not realize at the time that the consequence of their decision was a fault



it is usually considered that both mistakes and bad decisions are accidental, as long as they are not made with malicious objectives.

However, not all mistakes and bad decisions by nonmalicious persons are accidents. We introduce a further partitioning of nonmalicious human-made faults into

1) accidental faults, and 2) incompetence faults.

HOW TO RECOGNIZE INCOMPETENCE FAULTS?

Important when consequences that lead to economic losses or loss of human life.

- Malicious human-made faults are introduced with the malicious objective to alter the functioning of the system during use.

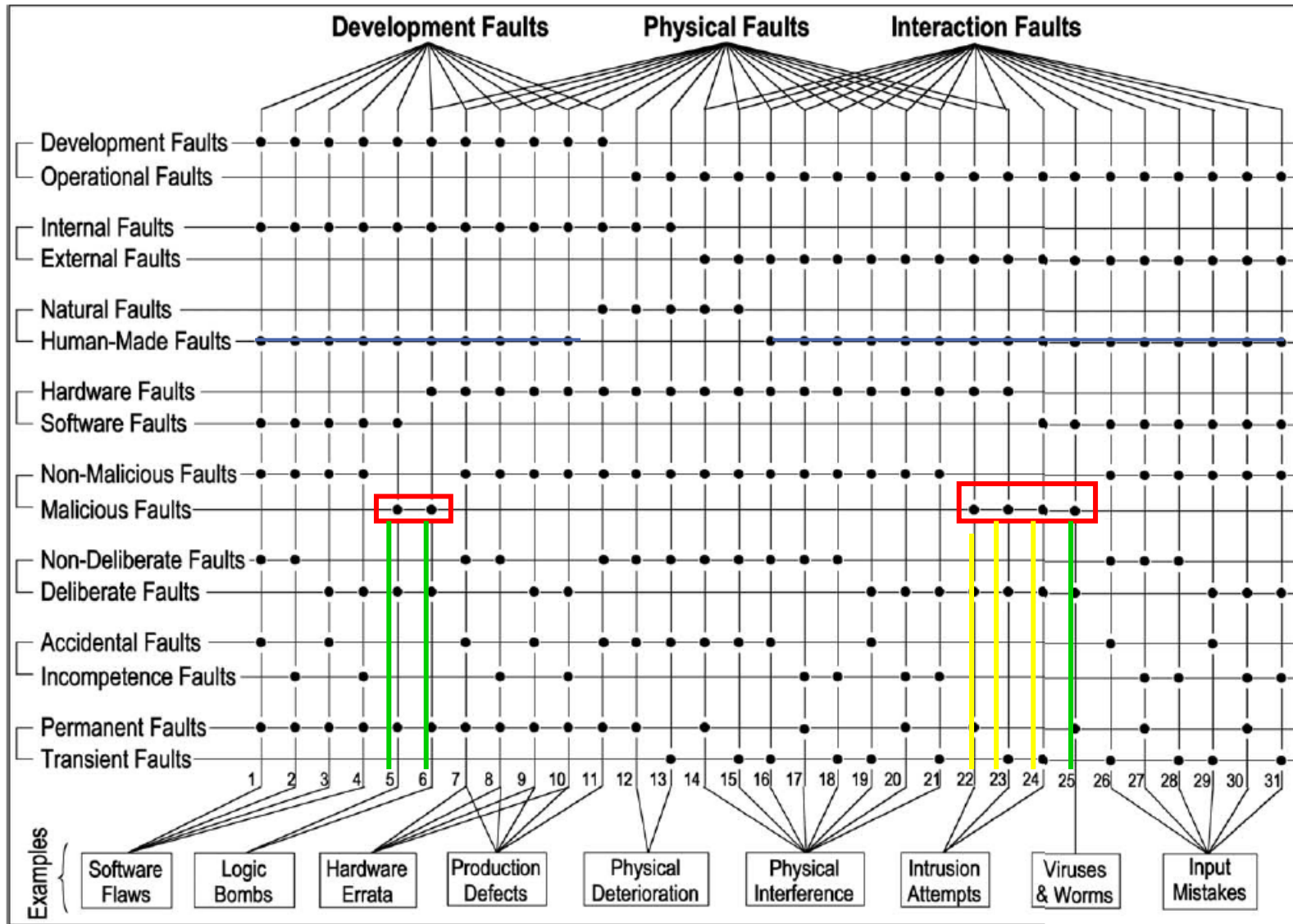
The goals of such faults are:

- to disrupt or halt service, causing denials of service;
- to access confidential information; or
- to improperly modify the system.

Malicious human-made faults are grouped into two classes:

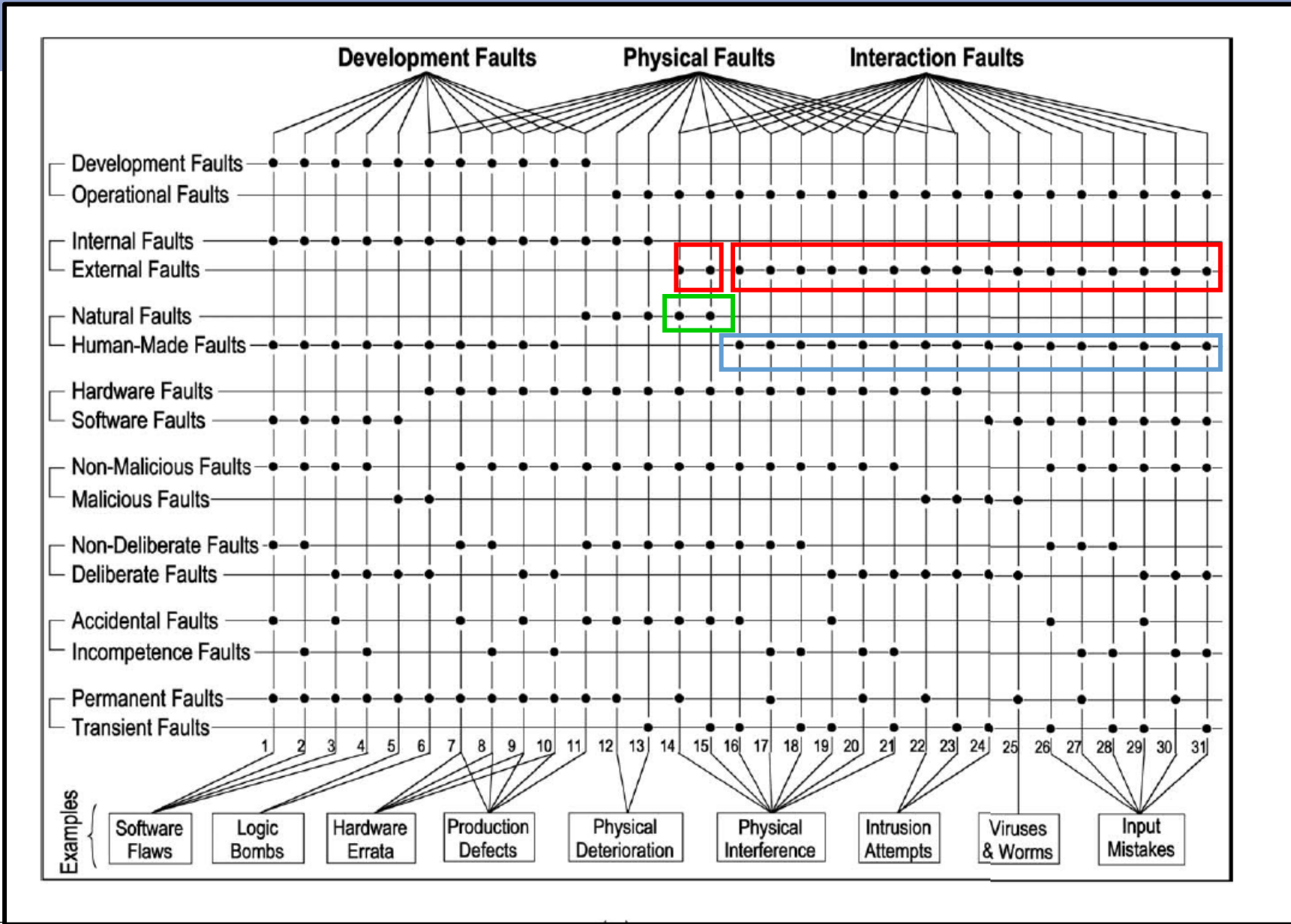
- Malicious logic faults that encompass development faults (5,6) such as Trojan horses, logic or timing bombs, and trapdoors, as well as operational faults (25) such as viruses, worms, or zombies.
- Intrusion attempts that are operational external faults (22-24). The external character of intrusion attempts does not exclude the possibility that they may be performed by system operators or administrators who are exceeding their rights,
Intrusion attempts may use physical means to cause faults: power fluctuation, radiation, wire-tapping, heating/cooling, etc.

Human-made Malicious faults



- Interaction faults occur during the use phase, therefore they are all operational faults. They are caused by elements of the use environment interacting with the system; therefore, they are all external. Most classes originate due to some human action in the use environment; therefore, they are human-made.
- They are fault classes 16-31. An exception are external natural faults (14-15) caused by cosmic rays, solar flares, etc. Here, nature interacts with the system without human participation.

Interaction faults



- A broad class of human-made operational faults are configuration faults, i.e., wrong setting of parameters that can affect security, networking, storage, middleware, etc.
- Such faults can occur during configuration changes performed during adaptive or augmentative maintenance performed concurrently with system operation (e.g., introduction of a new software version on a network server); they are then called reconfiguration faults.

- A common feature of interaction faults is that, in order to be “successful,” they usually necessitate the prior presence of a vulnerability, i.e., an internal fault that enables an external fault to harm the system.
- A vulnerability can result from a deliberate development fault, for economic or for usability reasons, thus resulting in limited protections, or even in their absence.

Permanent/Transient faults



- Permanent fault
a fault continuous and stable.
It remains in existence if no corrective action is taken.
- Transient fault a fault that can appear and disappear within a very short period of time



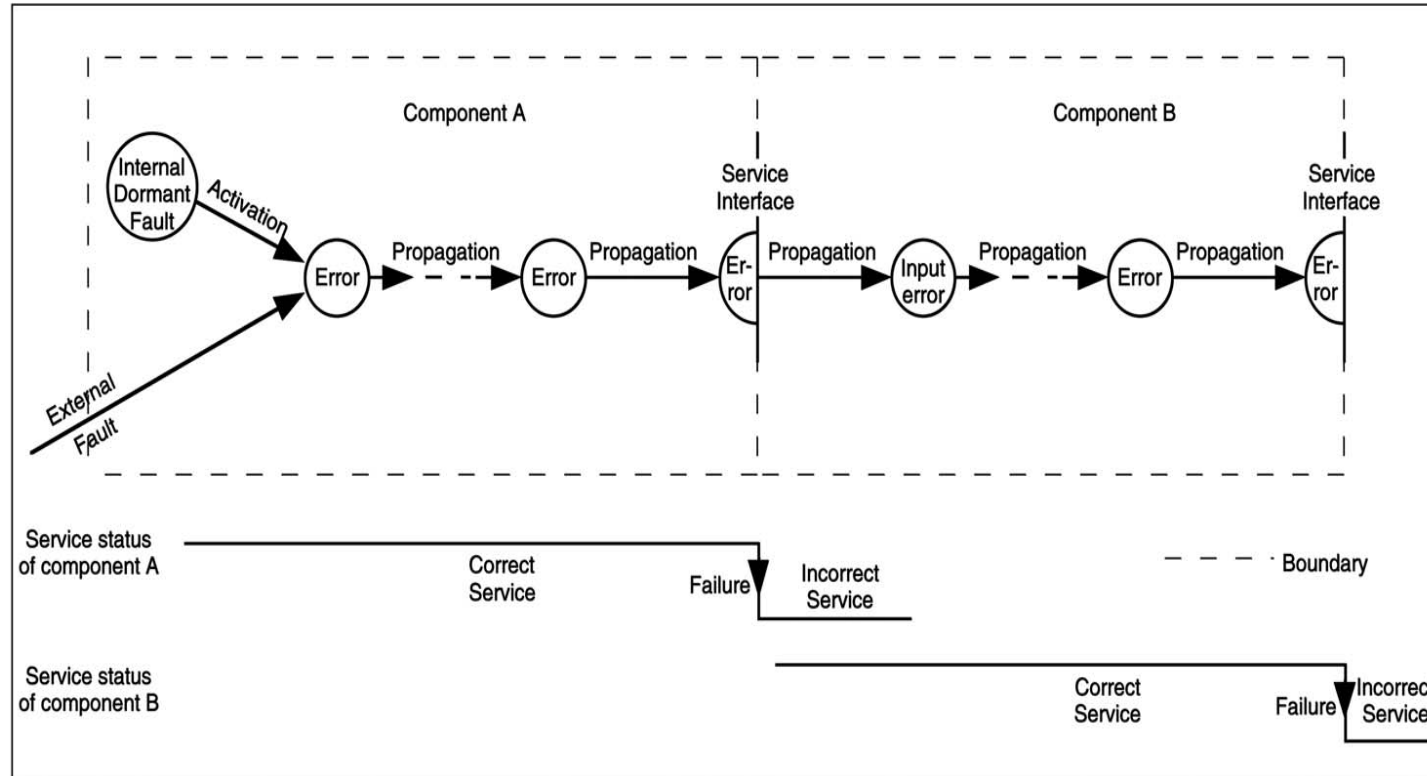
Errors

An error is **detected** if its presence is indicated by an error message or error signal. Errors that are present but not detected are **latent** errors.

Whether or not an error will actually lead to a service failure depends on two factors:

1. The structure of the system, and especially the nature of any redundancy that exists in it
2. The behavior of the system: the part of the state that contains an error may never be needed for service, or an error may be eliminated (e.g., when overwritten) before it leads to a failure.

Relationship Faults-Errors-Failures



From [Avizienis et al., 2004]

Example



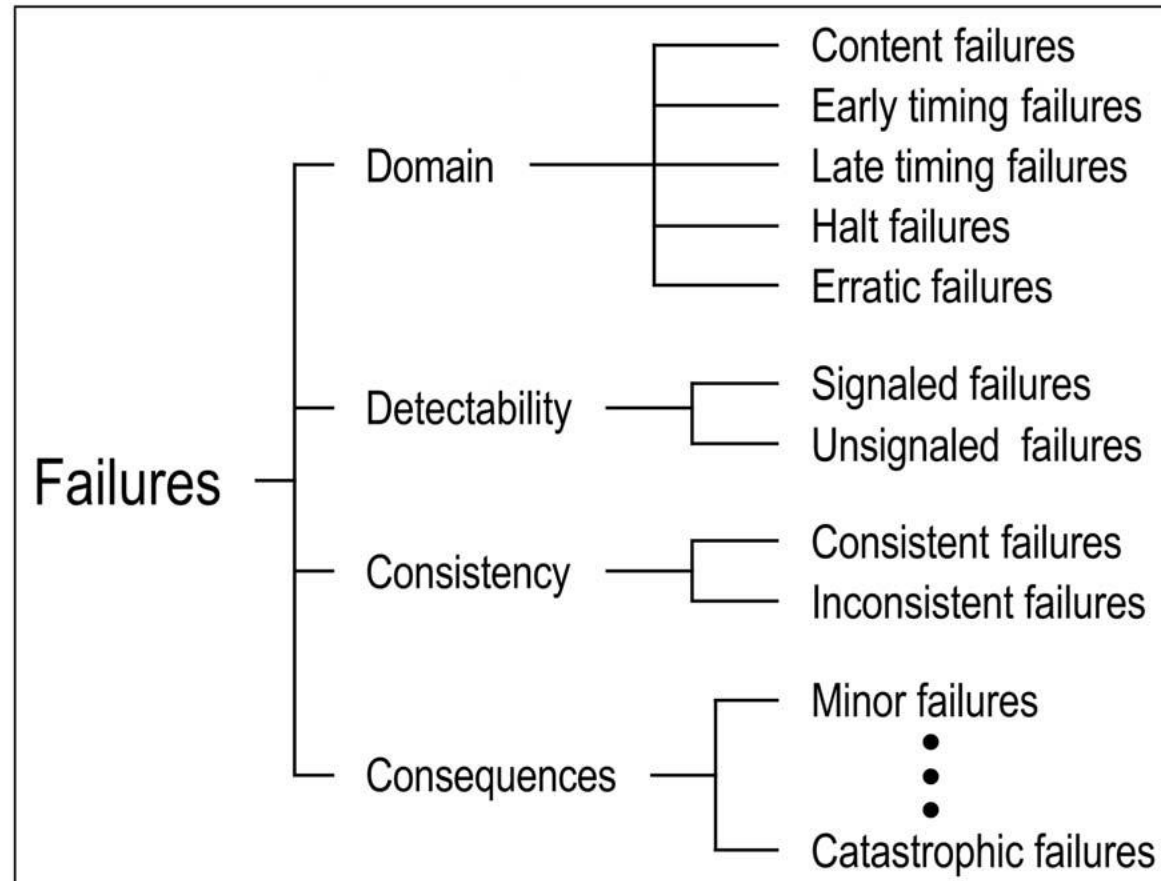
- Assume the sensor reporting the speed at which the main turbine is spinning breaks, and reports that the turbine is no longer spinning.
- The failure of the sensor injects a fault (incorrect data) into the system.
- This fault causes the system to send more steam than required to the turbine (error), thus over-speeding the turbine and activating the safety mechanism that shuts down the turbine to prevent damaging it.





Failures

Failures



From [Avizienis et al., 2004]

The service failure modes characterize incorrect service according to four viewpoints:

1. the failure domain,
2. the consistency of failures,
3. the detectability of failures and
4. the consequences of failures on the environment.

1. The **failure domain** viewpoint leads us to distinguish

- **content failures**
the content of the information delivered at the service interface (i.e., the service content) deviates from implementing the system function.
- **timing failures**
the time of arrival or the duration of the information delivered at the service interface (i.e., the timing of service delivery) deviates from implementing the system function.

2. The **consistency viewpoint** of failures leads us to distinguish, when a system has two or more users:

- consistent failures.
the incorrect service is perceived identically by all system users.
- inconsistent failures.
some or all system users perceive differently incorrect service (some users may actually perceive correct service); inconsistent failures are usually called, Byzantine failures.

3. The **detectability viewpoint** addresses the signaling of service failures to the user(s).

Signaling at the service interface originates from detecting mechanisms in the system that check the correctness of the delivered service.

- The detecting mechanisms themselves have two failure modes:
 - 1) signaling a loss of function when no failure has actually occurred, that is a false alarm,
 - 2) not signaling a function loss, that is an unsignaled failure.

4. Grading the **consequences** of the failures upon the system environment enables failure severities to be defined.

- minor failures
where the harmful consequences are of similar cost to the benefits provided by correct service delivery;
- catastrophic failures
where the cost of harmful consequences is orders of magnitude, or even incommensurably, higher than the benefit provided by correct service delivery.

Dependability tree

