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

Outline

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

Dependability: a definition

- A system is designed to provide a certain service
- Dependability is the ability of a system to deliver the specified 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”

Dependable computer-based systems

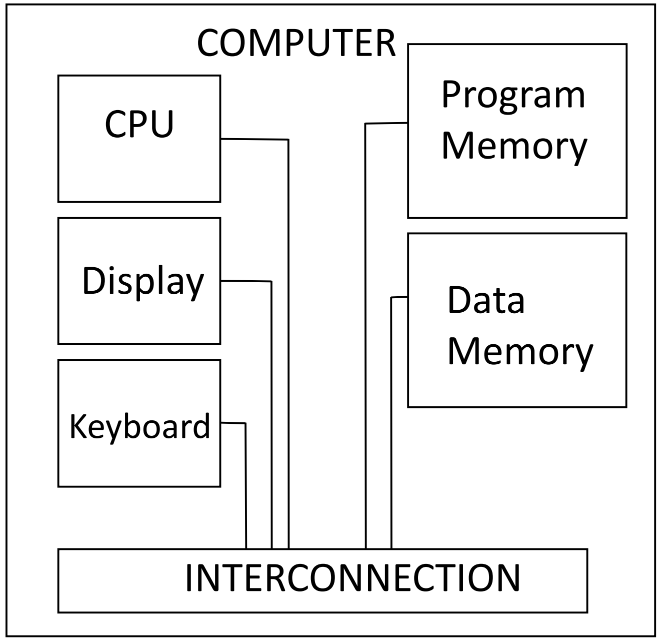
- Dependability is important in safety-critical systems, systems whose failure or malfunction may result in death or serious injury to people, loss or serious damage of equipment, or environmental harm.

- Future safety-critical systems will be more automated and more dependent on computers than today's systems

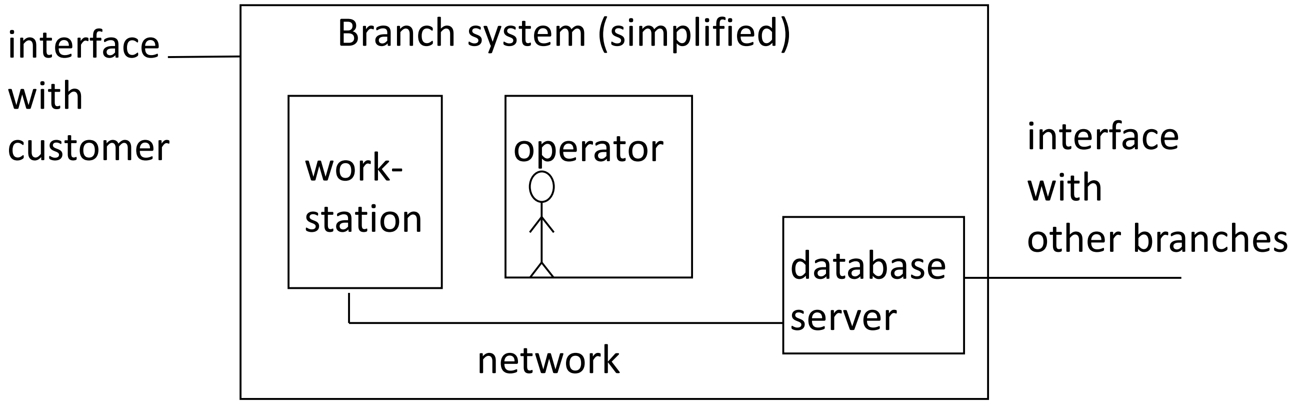
- Computers are increasingly used in safety-critical systems:
  - transport (automotive, railways, aerospace, …)
  - medicine
  - process control
  - ….

Systems and components

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



Bank information system



Failure, fault, error

- If the system stops delivering the intended service, we call this a failure The correct service may later restart.

For instance:

- deliver 200 Euro when you asked 20 Euro

- We call the causes of failures faults
- A fault causes an error in the state of the system
- The error causes the system fail

Computer-based systems failures

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-based systems failures

Computer failures differ from failures of other equipment

- 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

- Small hidden faults may have large effects (digital machine)

- Computing systems are complex hierarchies relaying on hidden components

Computer-based systems failures

System: entity that interacts with other entities, i.e., other systems

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

These other systems are the environment of the given system.

Moreover,

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

Threats to Dependability

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

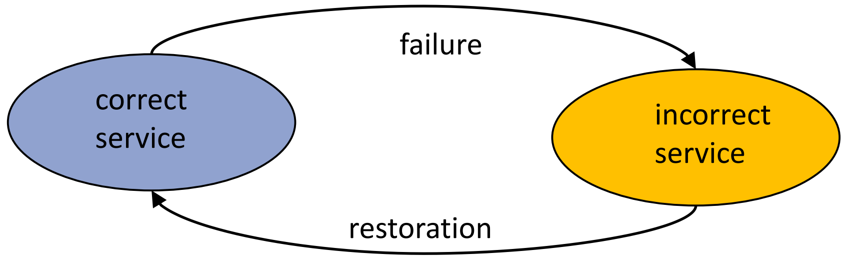
Failure
the external state is affected by the error

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.

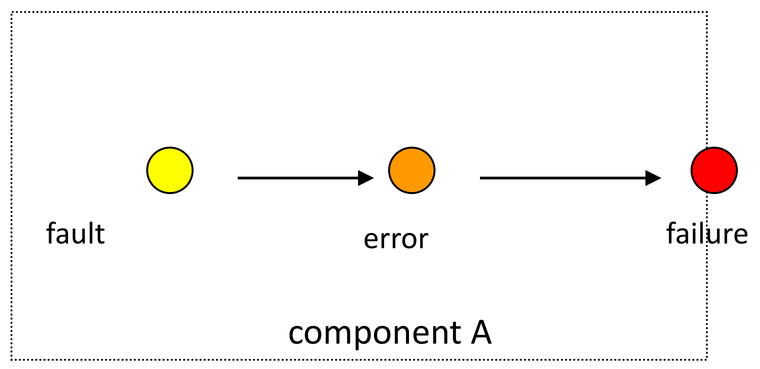
It is important to note that many errors do not reach the system’s external state and cause a failure.

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

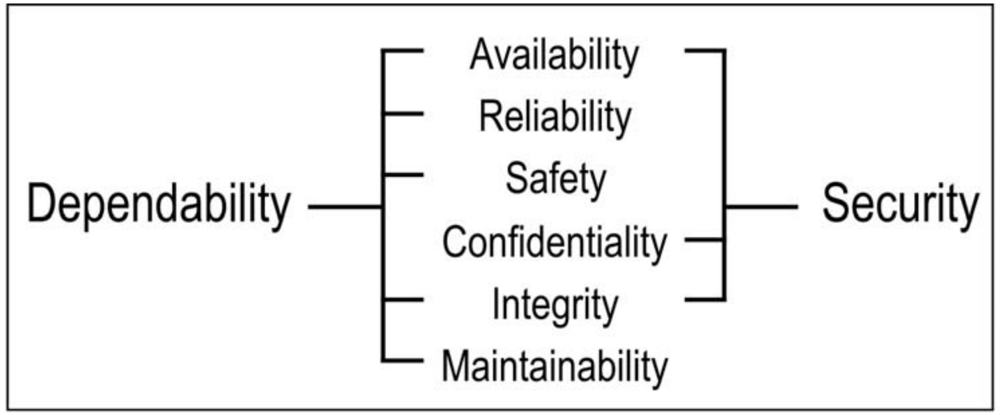


A fault causes an error in the internal state of the system.
The error causes the system to fail

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



From [Avizienis et al., 2004]

Dependability properties can be measured in terms of probability

Dependability attributes

- **Availability**
  readiness for correct service

- **Reliability**
  continuity of correct service

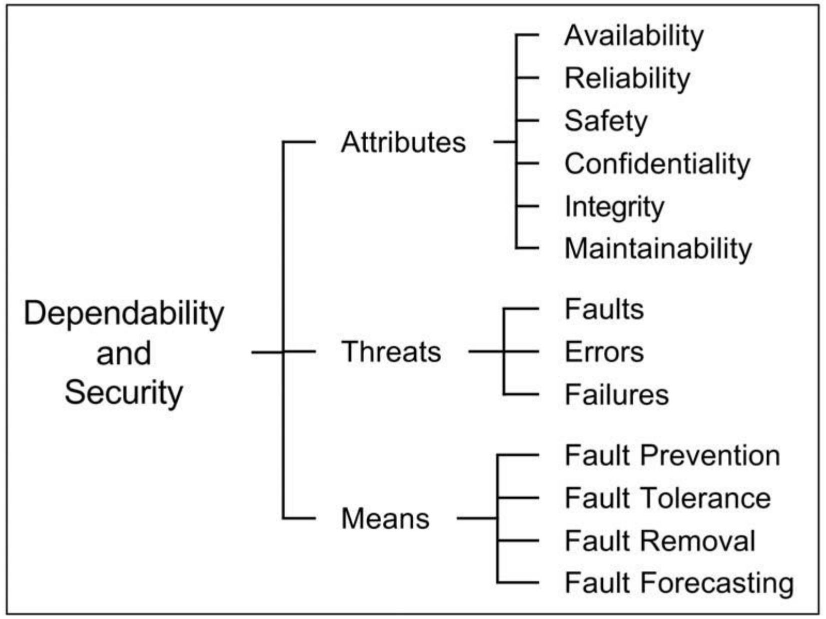
- **Safety**
  absence of catastrophic consequences on the user(s) and the environment

- **Confidentiality**
  the absence of unauthorized disclosure of information

- **Integrity**
  absence of improper system alterations

- **Maintainability**
  ability to undergo modifications and repairs

Dependability tree



From [Avizienis et al., 2004]

(*) Security: Availability, Confidentiality, Integrity

Threats to dependability

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

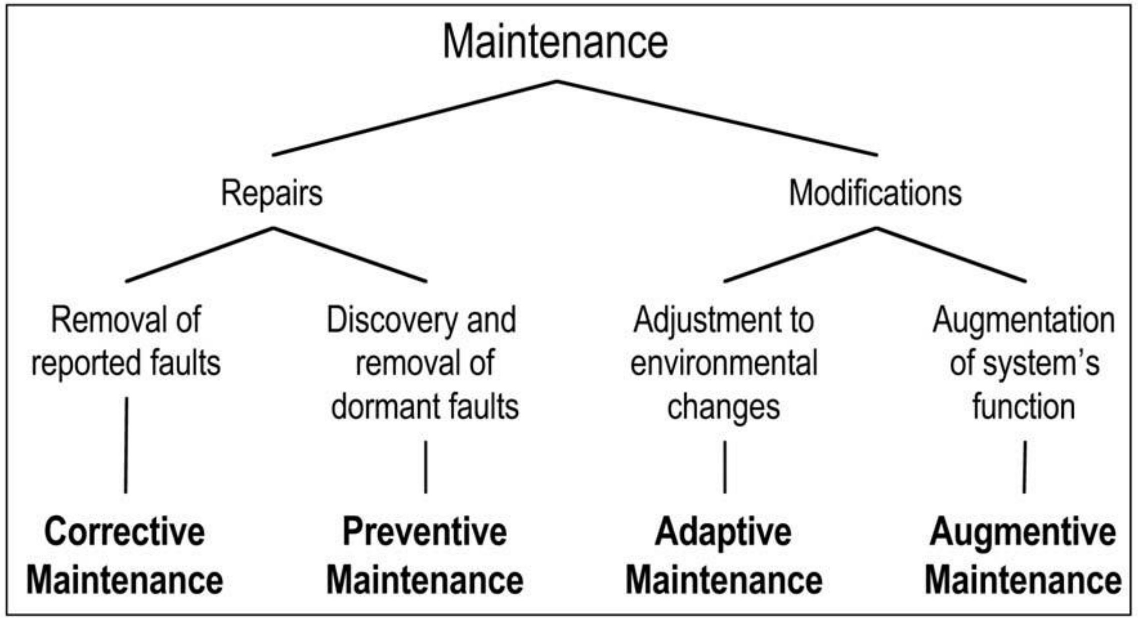
- The **use phase** consists of alternating periods of **correct service** delivery (to be called service delivery), **service outage**, and service **shutdown**.

- 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.

Maintenance


Maintenance
Repairs
Modifications
Removal of reported faults
Discovery and removal of dormant faults
Adjustment to environmental changes
Augmentation of system's function
Corrective Maintenance
Preventive Maintenance
Adaptive Maintenance
Augmentive Maintenance

From [Avizienis et al., 2004]

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

UNIVERSITÀ DI PISA

Faults

Faults classification

Identified combinations

three major partially overlapping groupings

- **Development faults**

  that include all fault classes occurring during development

- **Physical faults**

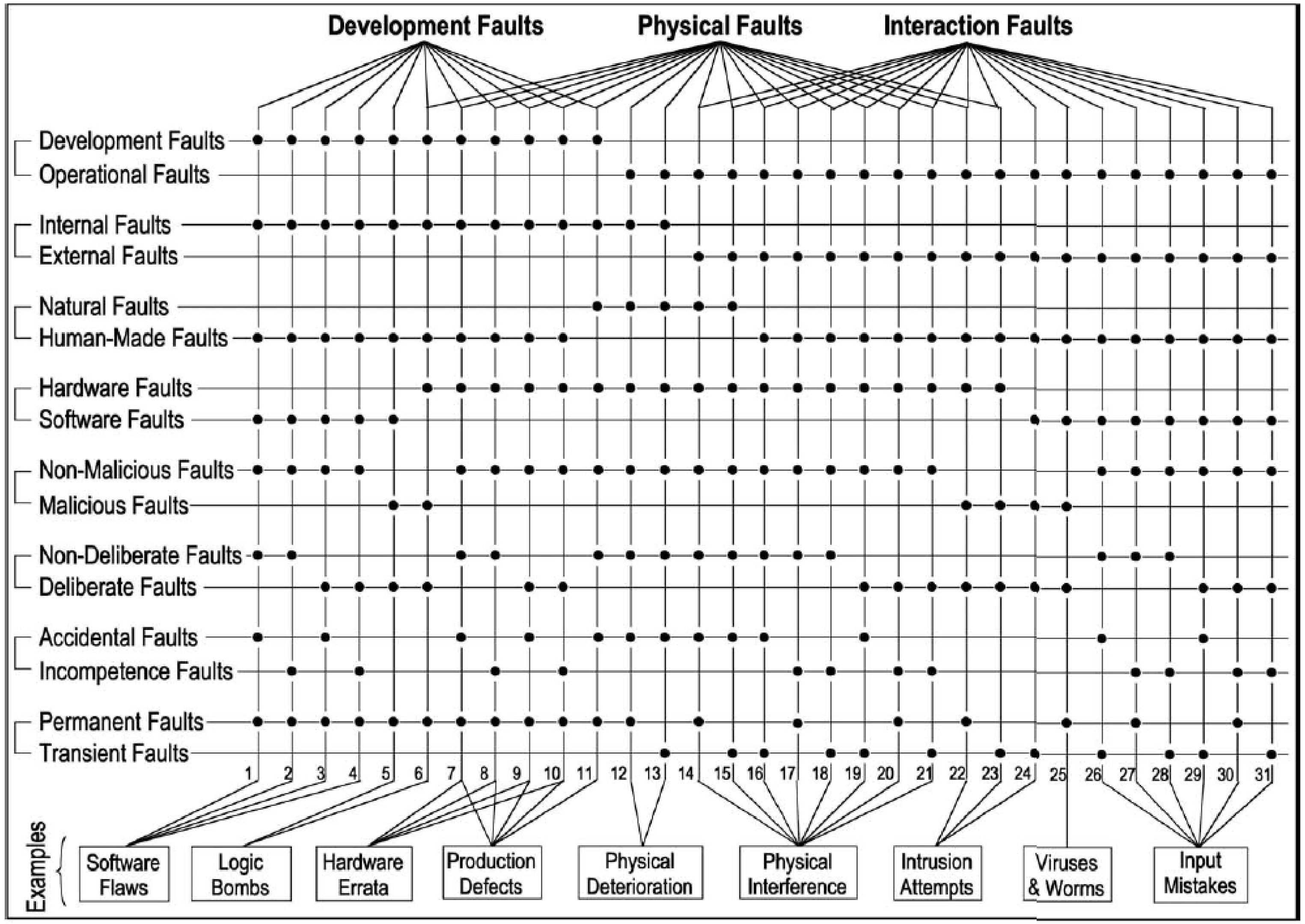
  that include all fault classes that affect hardware

- **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

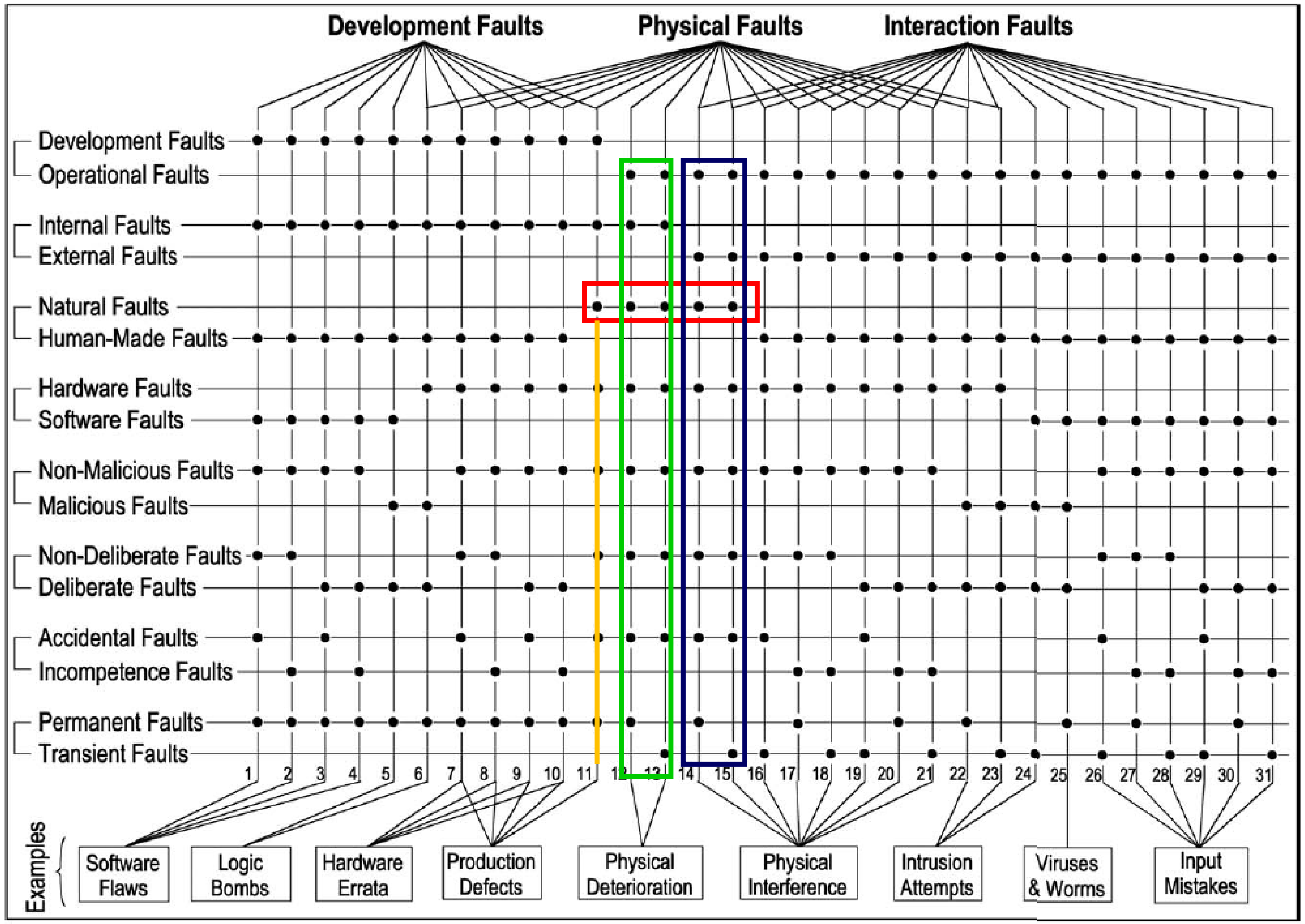
- Natural faults (11-15) are physical (hardware) faults that are caused by natural phenomena without human participation

- 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



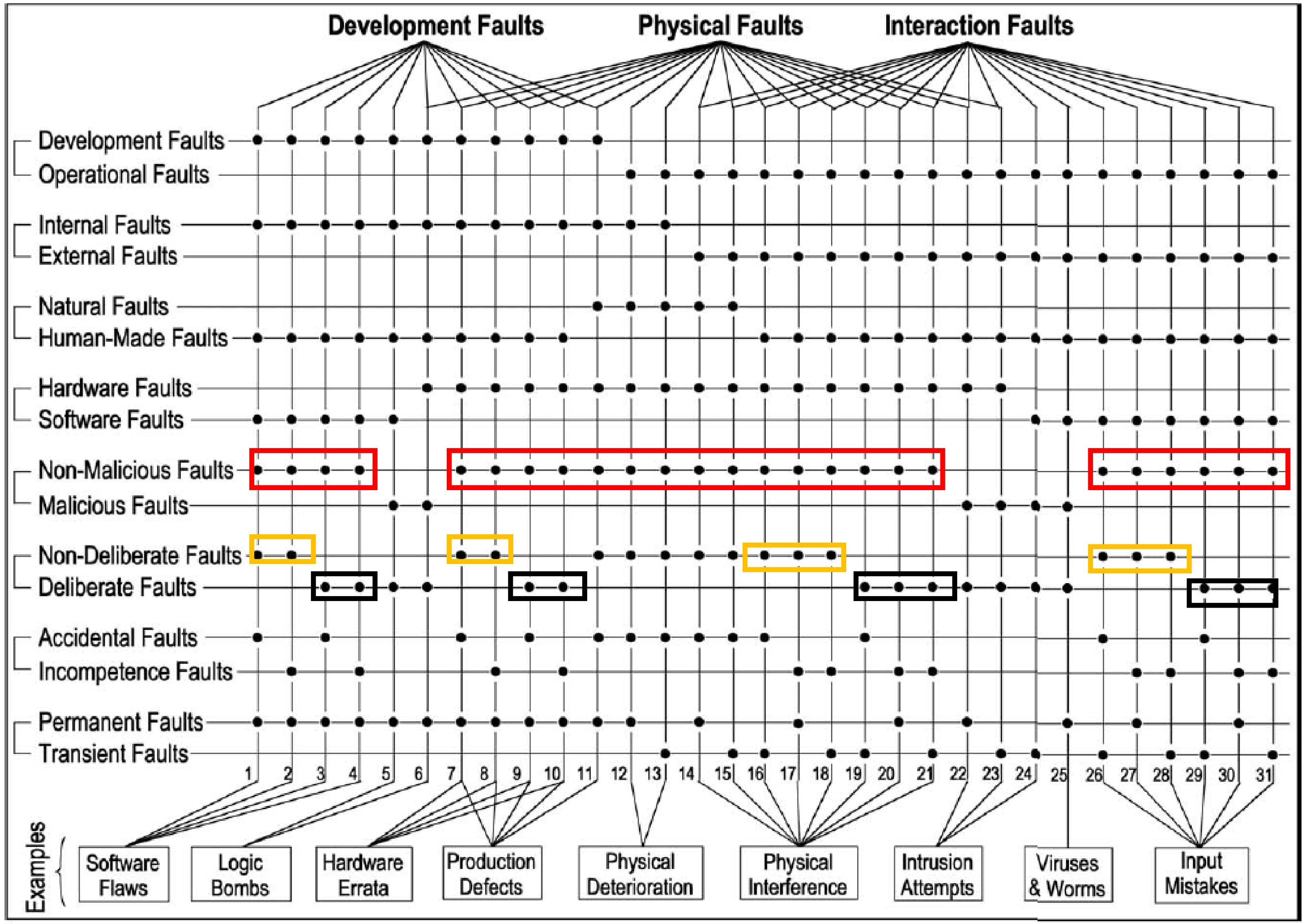
Human-Made Faults

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, 7-21, 26-31), introduced without malicious objectives.

Human-Made Faults



Human-Made 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 Human-Made Faults

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)

$\swarrow$ $\searrow$

development **interaction**

Non-malicious Human-Made Faults

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

Non-malicious Human-Made Faults

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.

Non-malicious Human-Made Faults

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

- 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

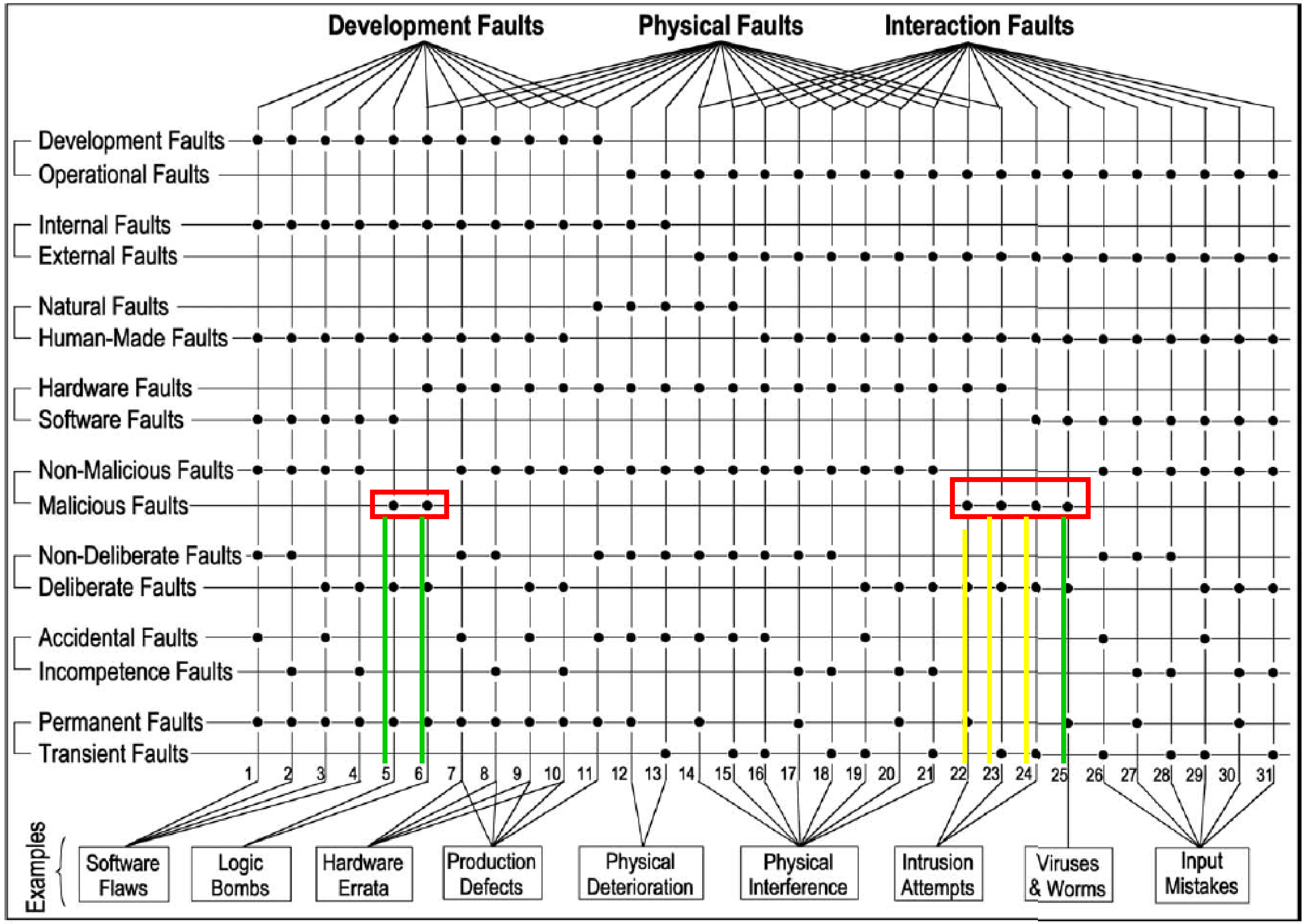
Malicious human-made faults are grouped into two classes:

- 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.

Malicious faults

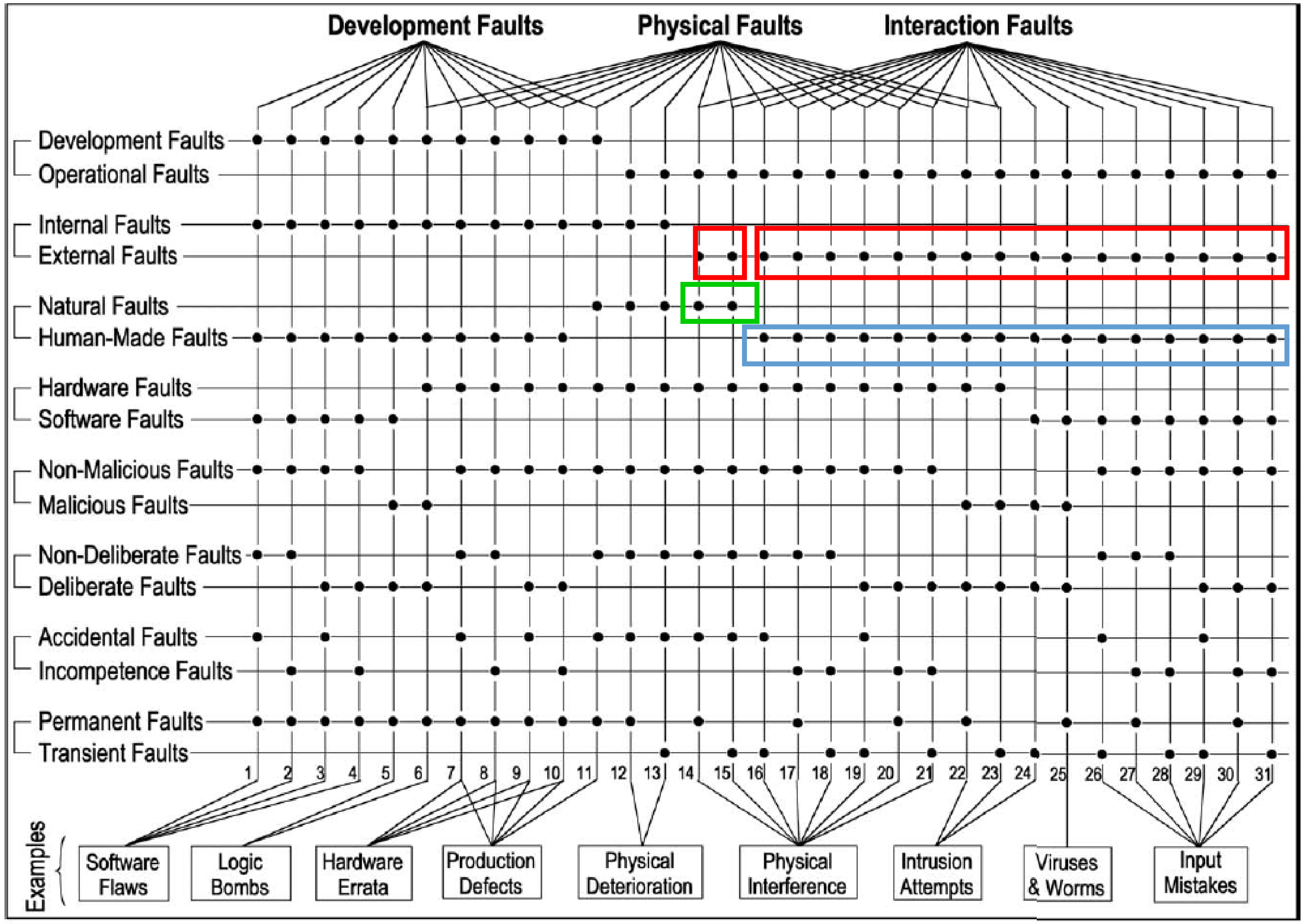


Interaction Faults

- 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



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.

Interaction 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.

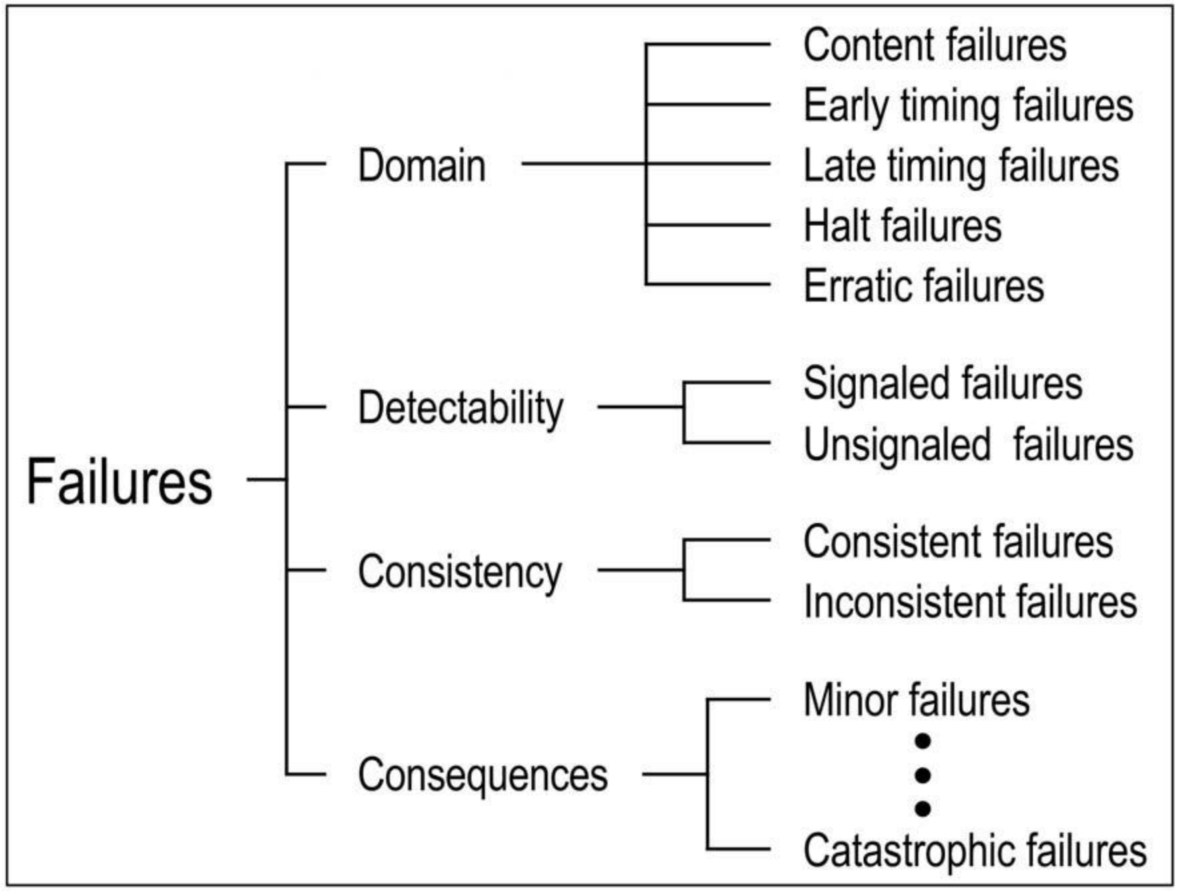
Permanent/Transient faults

- 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

Failures

Failures



From [Avizienis et al., 2004]

Failures

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.

Failures

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.

Failures

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.

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.

Failures

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.

Errors

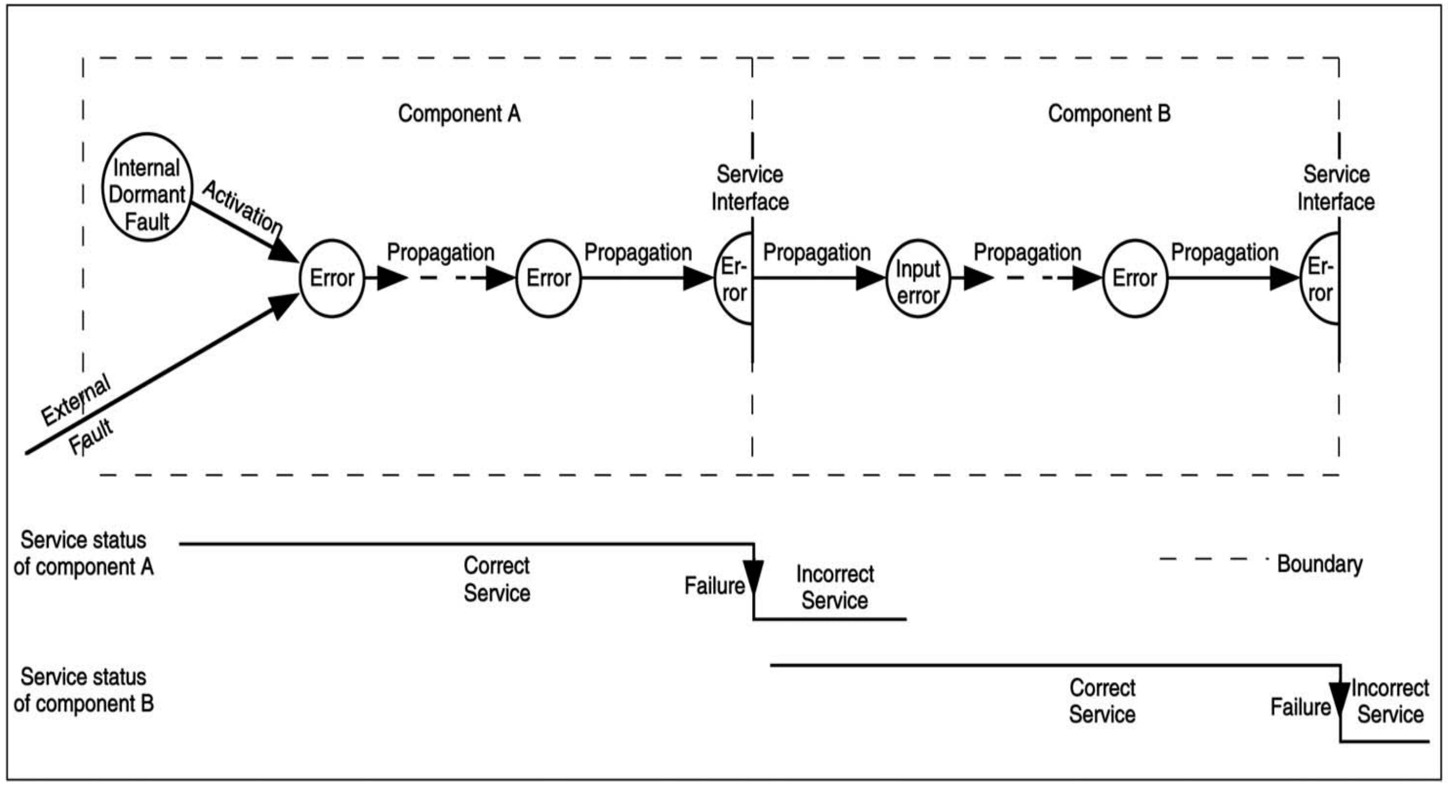
An error is detected if its presence is indicated by an error message or error signal. Errors that are present but not detected are latent errors.

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.

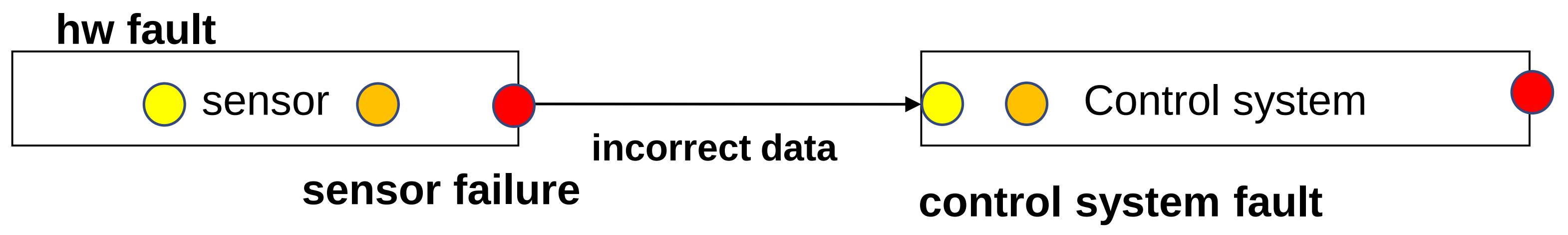
Relationship Faults-Errors-Failures



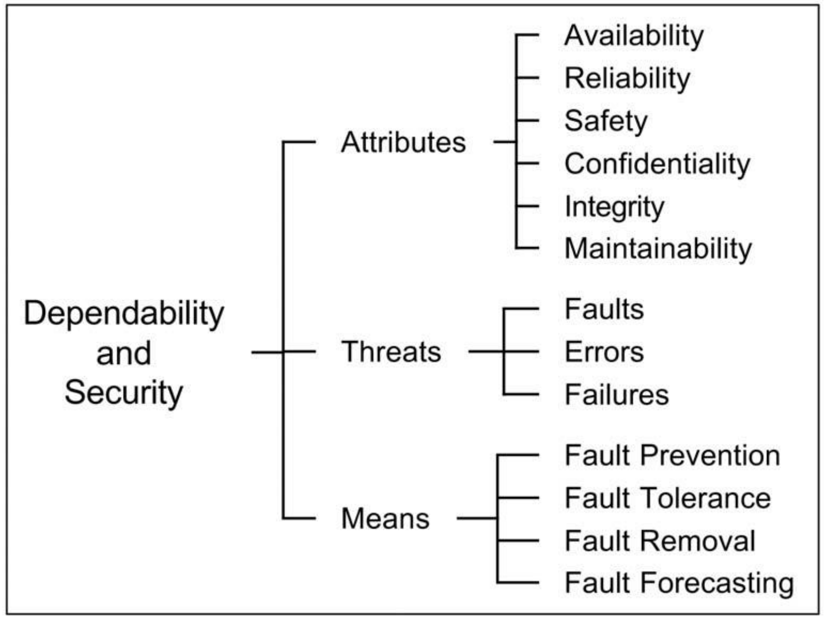
From [Avizienis et al., 2004]

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.



Dependability tree


Dependability and Security
Attributes
Availability
Reliability
Safety
Confidentiality
Integrity
Maintainability
Threats
Faults
Errors
Failures
Means
Fault Prevention
Fault Tolerance
Fault Removal
Fault Forecasting