Dependability: Concepts and Definitions

Andrea Bondavalli

Resilient Computing Lab
Dipartimento di Matematica e Informatica
Università degli Studi di Firenze

Università degli Studi di Firenze

- 60.000 students, 2500 foreigners
- 12 faculties, more than 150 degree courses
- 2.300 professors and researchers

- 750 research fellows
- 100 temporary researchers
- 1.400 PhD students
- 1.700 technicians and administrative people

The RCL Group is part of the

Dipartimento di Matematica ed Informatica

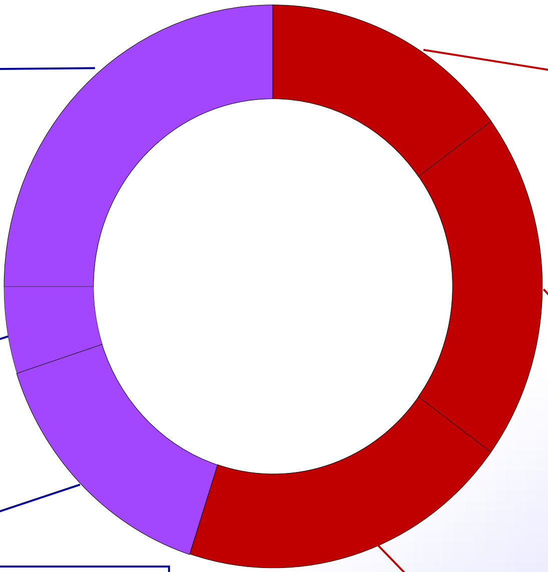
(DiMaI)
Viale Morgagni, 65
50134 – Firenze , Italy

http://www.dimai.unifi.it/

Resilient Computing Lab Research Activities

Architectures and techniques for resilient systems, infrastructures and networks

Validation of systems dependability, trust and QoS through analytical, simulative and experimental techniques



Generic architectures for fault-tolerant and safe real-time systems

Security (Intrusion detection and tolerance)

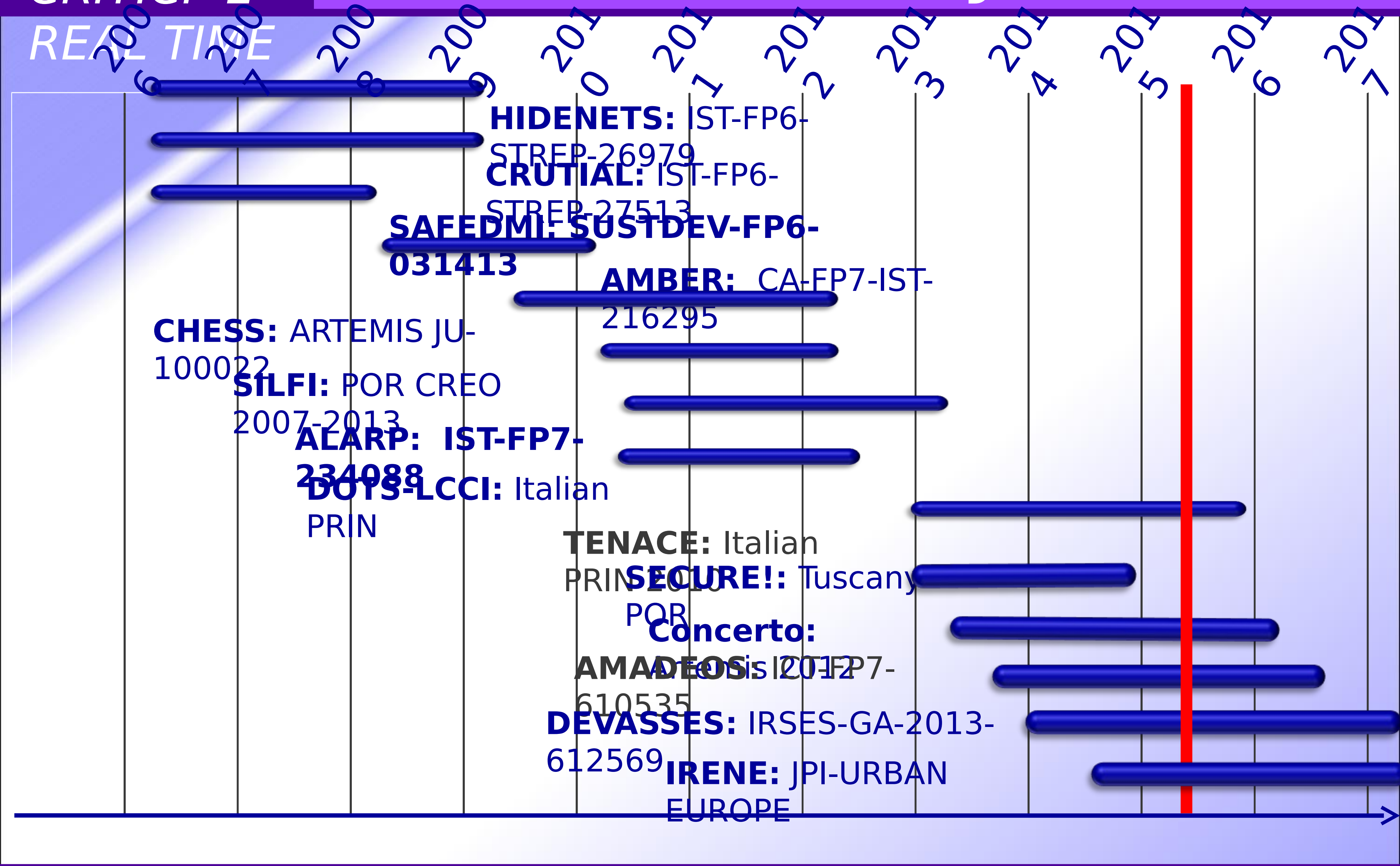
Online Diagnosis and dynamic reconfiguration in resilient and highly adaptive and heterogeneous systems

Simulative and analytical modelling & methodologies

Experimental evaluation , Robustness testing and Fault injection

Quality of Service in Network Systems and Protocols

Main Research Projects



Introduction: Concepts and Definitions

- Basic Dependability Concepts,
- Definitions:
- Faults, Errors and Failures
- Dependability Attributes
- Means to attain dependability
  - Fault Prevention,
  - Fault Tolerance,
  - Fault Removal,
  - Fault Forecasting.

Motivations

- Our society depends more and more on proper behaviour of computing systems.
- They are used for more and more critical applications very often overcoming human intervention (either because of an explicit design choice or because humans have unacceptable reaction time).
- Examples of critical applications:
  - nuclear power stations
  - aeroplanes
  - railway interlocking
  - chemical plants
  - space shuttles
  - telephone networks

What is a system?

- Many things, but in our context, a collection of
  - hardware
  - networks
  - operating systems, and
  - application software

  that is intended to be dependable, secure, survivable or have predictable performance.
- To deal with such systems we must review
  - Basic performance and dependability concepts and measures
  - Fault/Error types
  - Fault prevention, removal and tolerance techniques

Some famous computer failures

- some examples over the years:
  - Radiation therapy machine overdoses patients
  - Space Shuttle can’t launch: synchronisation error in redundant computers
  - Buggy recovery software interrupts long-distance telephone service for hours in many states of U.S.A.
  - One wrong key stroke dooms Russian satellite
  - Ariane 5 lost due to *correct* component from Ariane 4
  - bank software sends avalanche of spurious order corrections, freezes part of NYSE affecting trading in 975 stocks
- ... continuing: e.g. in 2009-2010:
  - (2009) Hospital software glitch causes incorrect medication dosages
  - (2010) Toyota to recall 400,000 Prius cars over perceived software problem affecting braking

Computer failures differ from failures of other

- Subtler failures than “breaking down”, “stopping working”, “ seizing”
- The computer is used to store information: there are many ways information can be wrong, many different effects both within and outside the computer
- Transient faults have permanent effect
- Small, hidden faults may have large effects (digital machines -> discontinuous behaviour)
- Computing systems are complex design hierarchies relying on hidden components

Dependability

- Computing systems are characterized by five fundamental properties: functionality, usability, performance, cost, and dependability.
- *"Dependability of a computing system is the ability to deliver service that can justifiably be trusted"*

- The service delivered by a system is its behavior as it is perceived by its user(s); a user is another system (physical, human) that interacts with the former at the service interface.

Extracted from: Avizienis, A.; Laprie, J.-C.; Randell, B.; Landwehr, C. "Basic concepts and taxonomy of dependable and secure computing", IEEE TDSC, Vol. 1 Page(s): 11- 33, 2004

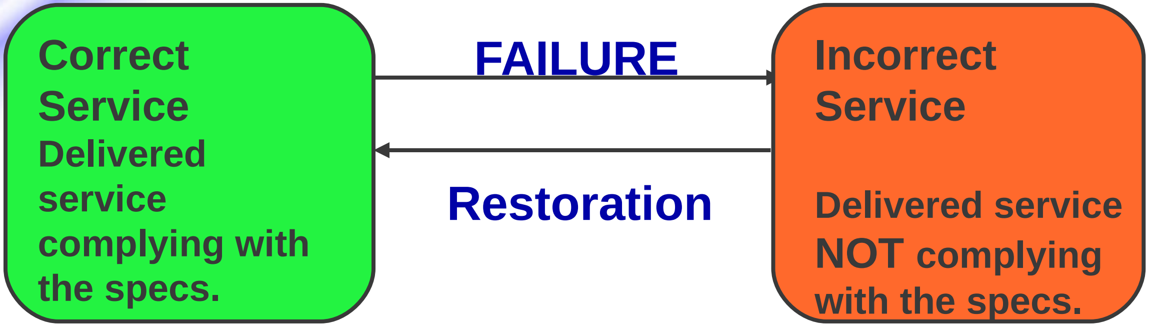
Definitions

- The function of a system is what the system is intended to do, and is described by the functional specification.
- Real-time service: function that must be provided within time intervals dictated by the environment
- Specification description of what is expected that the systems does, in terms of:
  - a) expected service;
  - b) conditions under which the service must be provided.

The service is usually specified in terms what is expected that the systems should do. Sometime the specification may be completed stating what the system should not do.

It must be agreed upon between the system provider and user and constitutes the basis for an objective judgment on whether the service is acceptable or a failure occurred.

Definitions


Correct Service
Delivered service complying with the specs.
FAILURE
Restoration
Incorrect Service
Delivered service NOT complying with the specs.

- Correct service is delivered when the service implements the system function.
- A system failure is an event that occurs when the delivered service deviates from correct service.
- A failure is thus a transition from correct service to incorrect service, i.e., to not implementing the system function.
- The delivery of incorrect service is a system outage.
- A transition from incorrect service to correct service is service restoration.

A systematic exposition of the concepts of dependability

Dependability		
Attributes	**Means**	**Impairments (Threats)**
Properties expected from the system and according to which assessment of service quality resulting from threats and means opposing to them is conducted	Methods and techniques enabling 1) to provide service on which reliance can be placed and 2) to have confidence in its ability	Undesired (not unexpected) circumstances causing or resulting from undependability (reliance cannot or will not any longer be placed on the service

Dependability Concepts

- *Measures* - properties expected from a dependable system
  - Availability
  - Reliability
  - Safety
  - Confidentiality
  - Integrity
  - Maintainability
  - Coverage

- *Means* - methods to achieve dependability
  - Fault Avoidance
  - Fault Tolerance
  - Fault Removal
  - Dependability Assessment

- *Impairments* - causes of undependable operation
  - Faults
  - Errors
  - Failures

The Threats: Faults, Errors and Failures

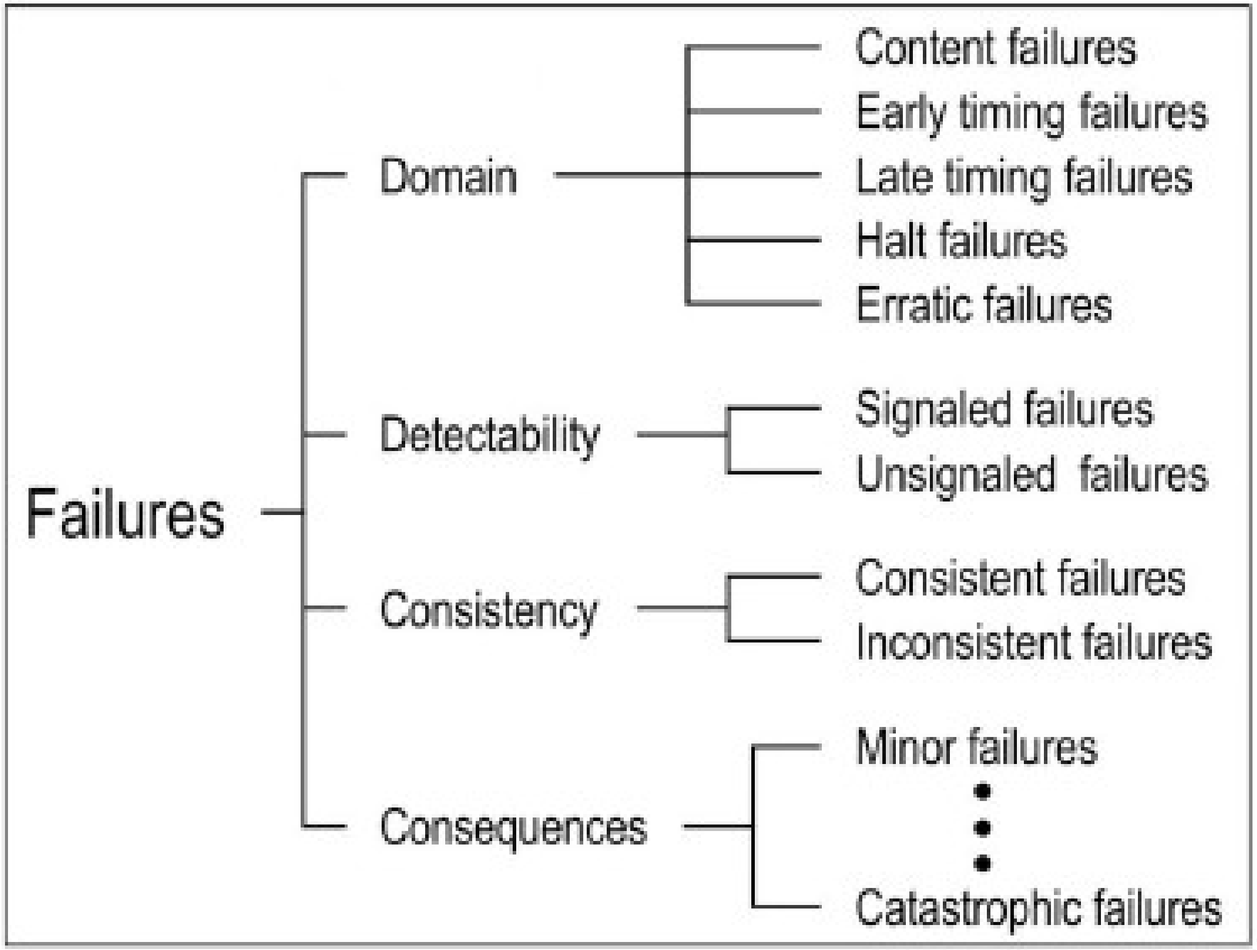
- A system may fail either because
  - it does not comply with the specification, or
  - the specification did not adequately describe its function.

- Error: that part of the system state that may cause a subsequent failure
- A Failure occurs when an error reaches the service interface and alters the service.
- Fault: the adjudged or hypothesized cause of an error. A fault is *active* when it produces an error; otherwise it is *dormant*.

Failures

- A system does not always fail in the same way: many failure modes.
- The modes characterize incorrect service according to four viewpoints:
  - the failure domain,
  - the controllability of failures,
  - the consistency of failures, when a system has two or more users,
  - the consequences of failures on the environment.

Failure modes



- *Service Failure Modes*

Errors and system state

- A system is a set of interacting components, thus
  - the system state is the set of its component states.
- A fault causes an error within the state of one (or more) components,
- but system failure will not occur until the error reaches the service interface of the system

Errors and system state - 2

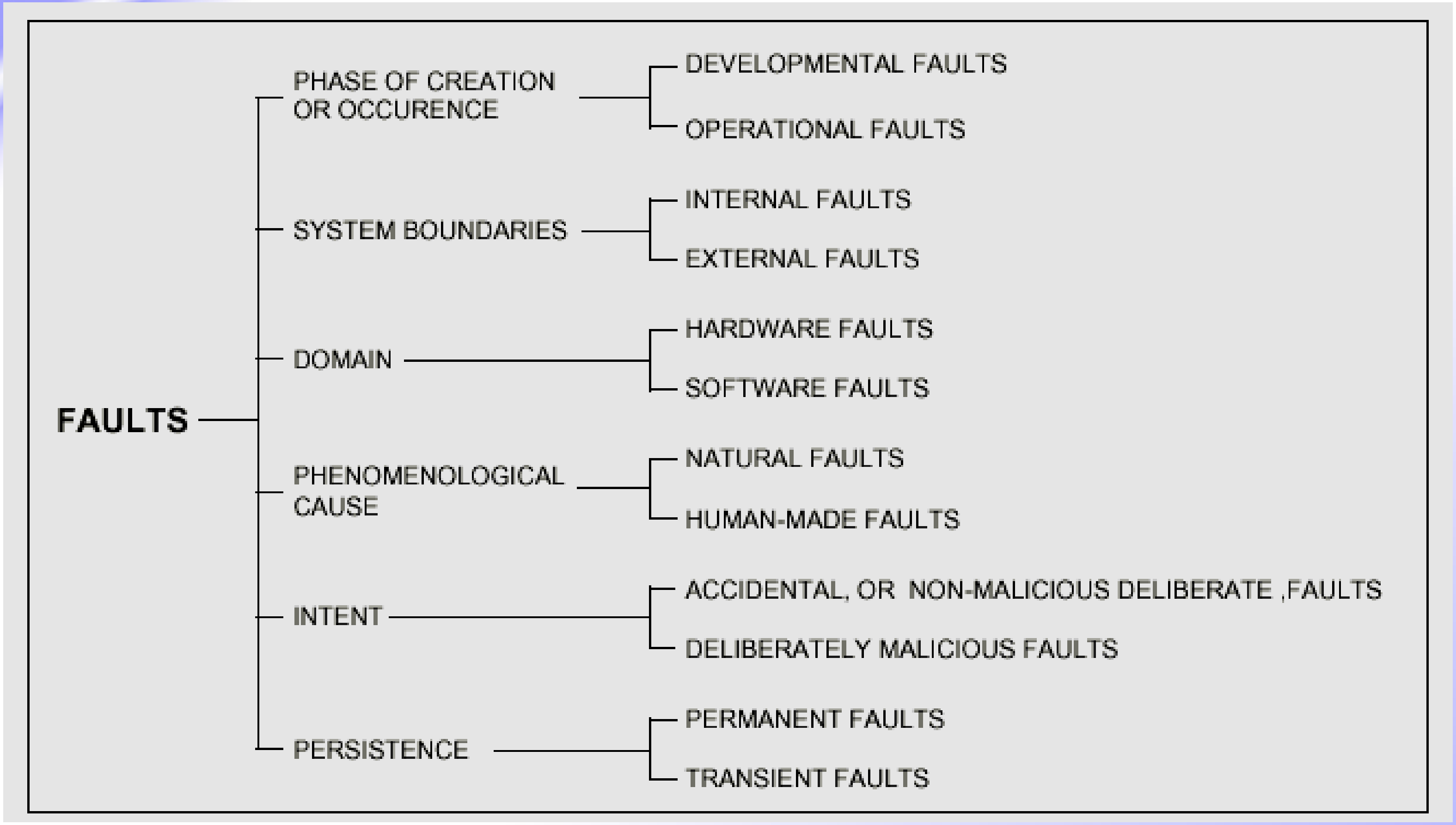
Classification of errors in terms of

- the *type* of component failures that they cause:
  - value vs. timing errors
  - consistent vs. inconsistent (‘Byzantine’) errors when the output goes to two or more components
- the *severity* of component failures that they cause:
  - minor vs. ordinary vs. catastrophic

- 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.

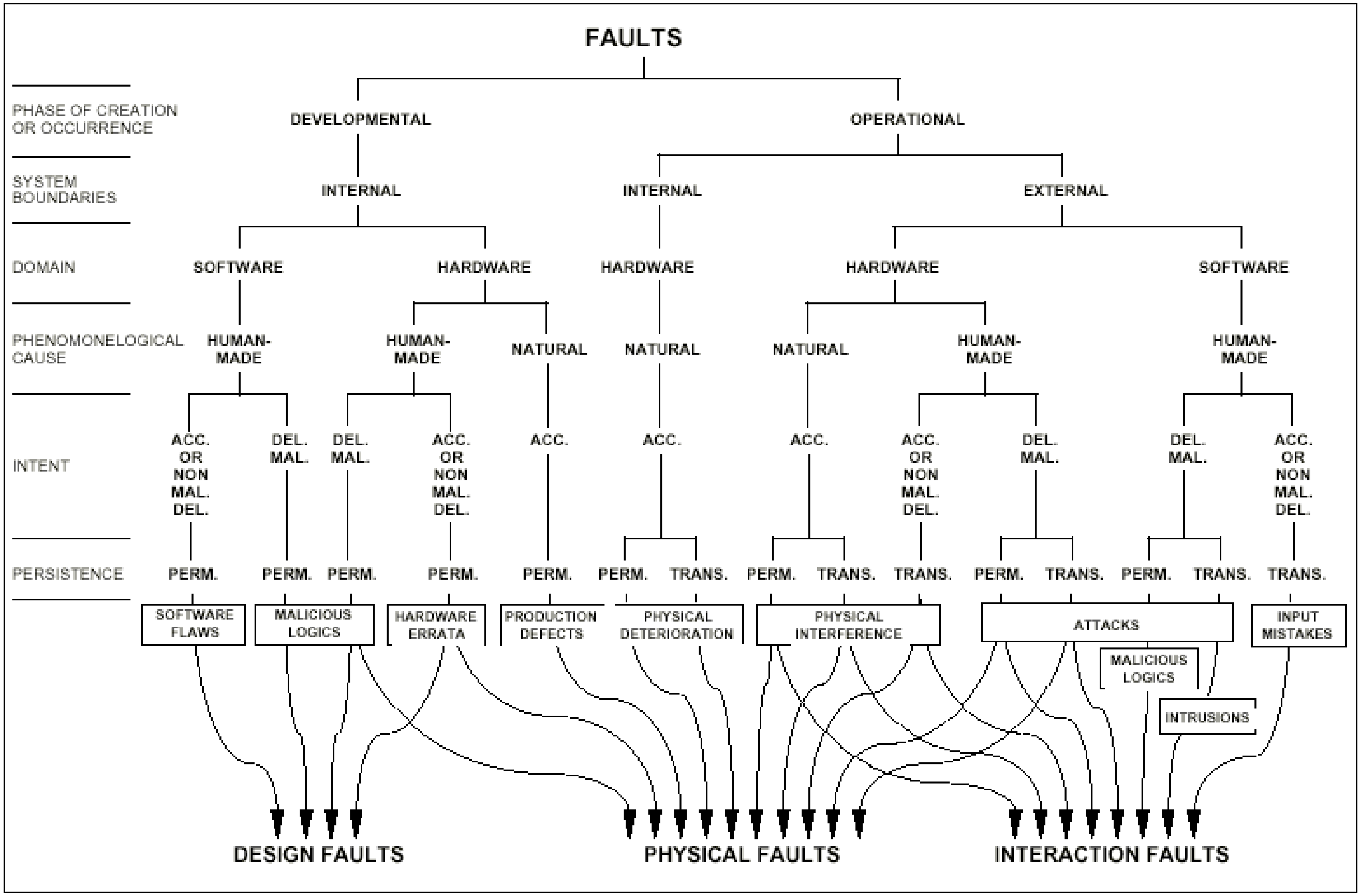
Faults

Faults are the adjudged or hypothesized cause of an error.



Elementary fault classes

design faults, physical faults, interaction faults,



Fault error failure chain - 1

- A fault is active when it produces an error, otherwise it is dormant.
- An active fault is either
  - an internal fault that was previously dormant, or
  - an external fault.
- Fault activation is the application of an input to a component that causes a dormant fault to become active.
- Error propagation within a component (internal propagation) is caused by the computation process: an error is successively transformed into other errors.

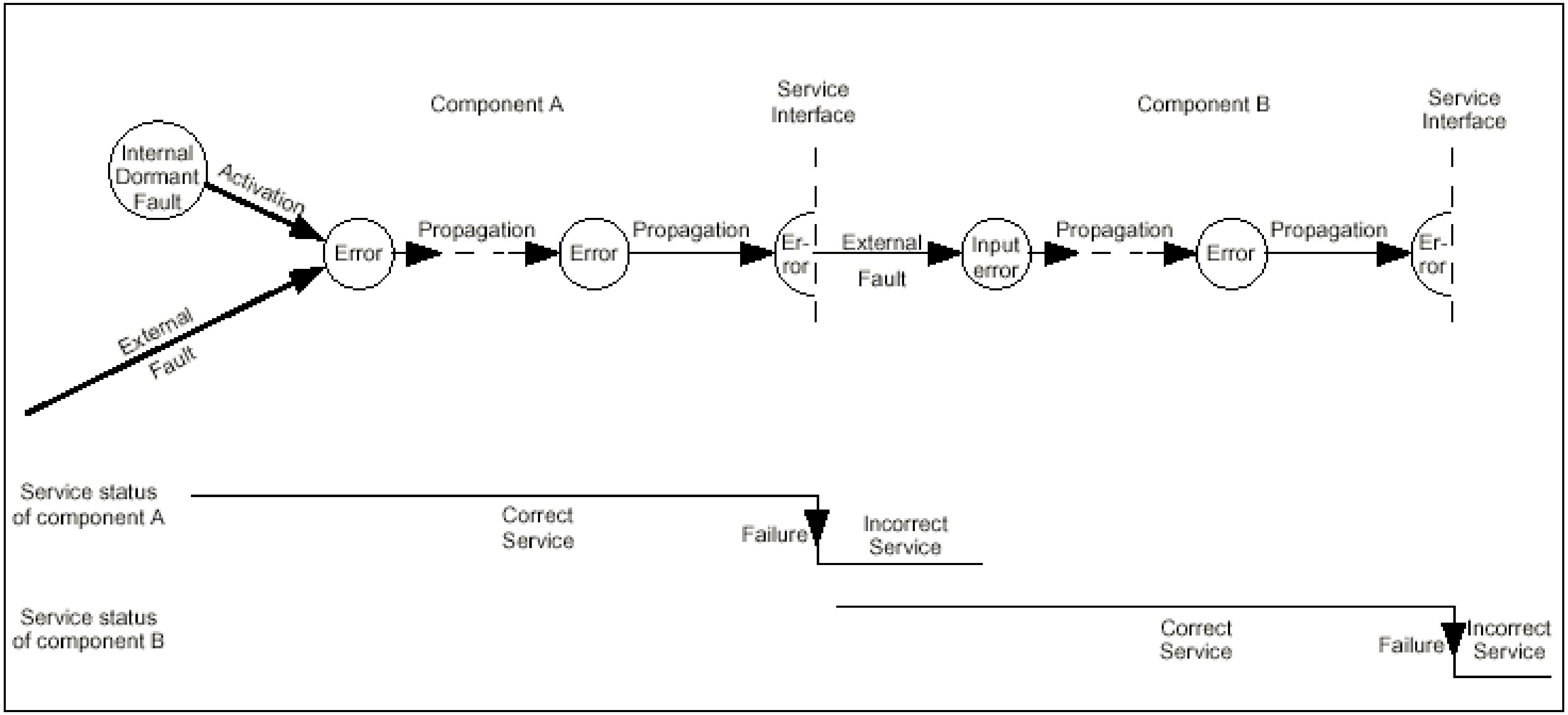
Fault error failure chain - 2

- Error propagation from one component (A) to another (B) that receives service from A (external propagation) occurs when an error reaches the service interface of component A (through internal propagation).
- At this time, A fails, and the failure of A appears as an external fault to B and propagates the error into B.

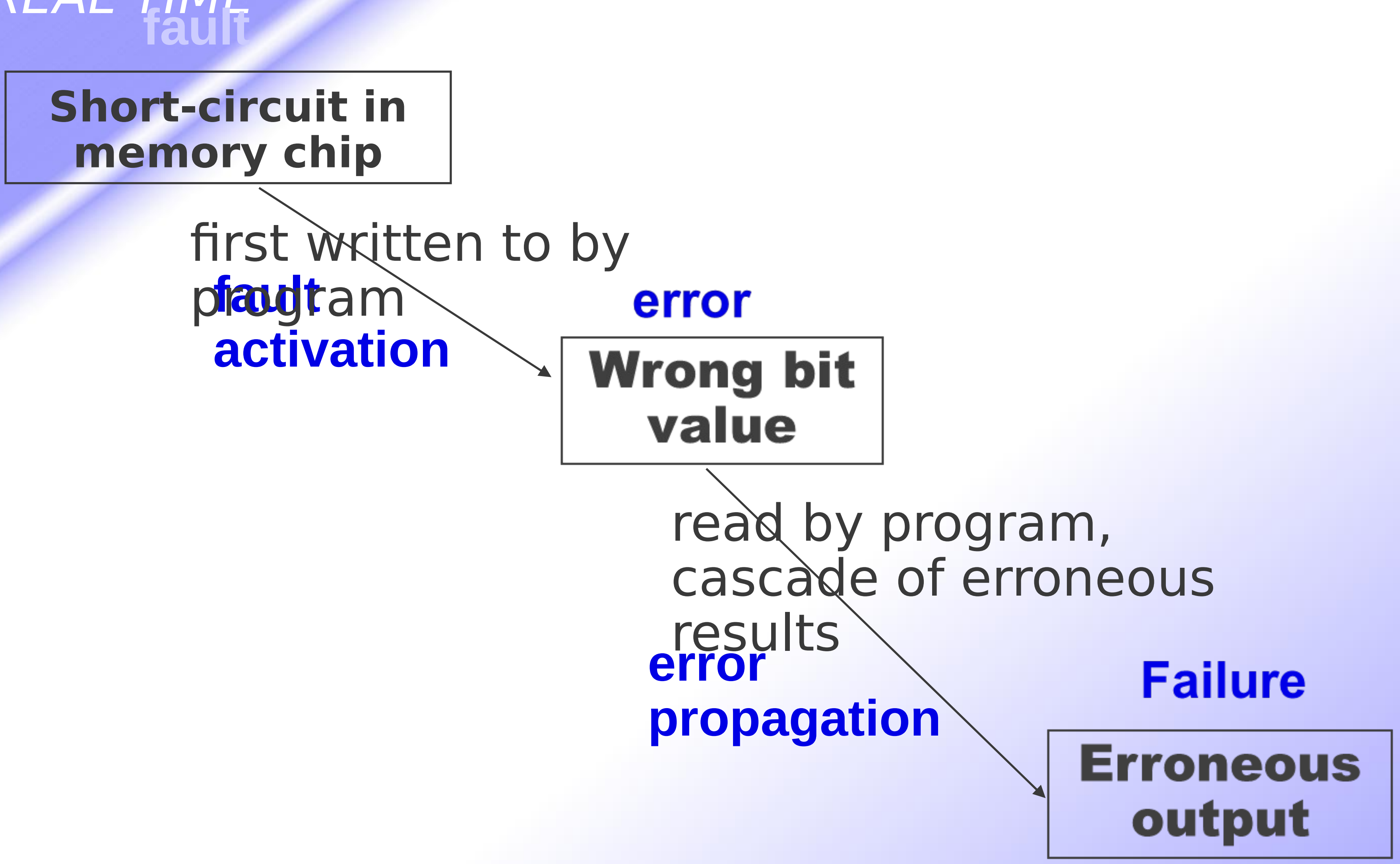
- A failure occurs when an error is propagated to the service interface and unacceptably alters the service delivered.
- A failure of a component causes a permanent or transient fault in the system that contains the component.

- Failure of a system causes a permanent or transient external fault for the other system(s) that interact with the given system.

Fault error failure chain - 3



Examples of faults-errors-failures


fault
Short-circuit in memory chip
first written to by program
fault activation
error
Wrong bit value
read by program, cascade of erroneous results
error propagation
Failure
Erroneous output

Types of Faults

- Failures may have many different causes, or *faults* :
  - chip suffers permanent electrical damage
  - undersized fan *(design fault)* allows overheating on hot days: chips malfunction *(physical fault)*; the machine works OK after cooling down (the fault is *transient*)
  - operator pushes the wrong button
  - one of two fans “burns out” (physical fault), repairman switches off the wrong fan (*maintenance* fault)
  - cosmic ray particle causing *transient* “upset” in execution
  - defect in software
  - software virus
  - a security *vulnerability* (software design fault), such that an attacker’s action (“interaction fault”?) causes failure

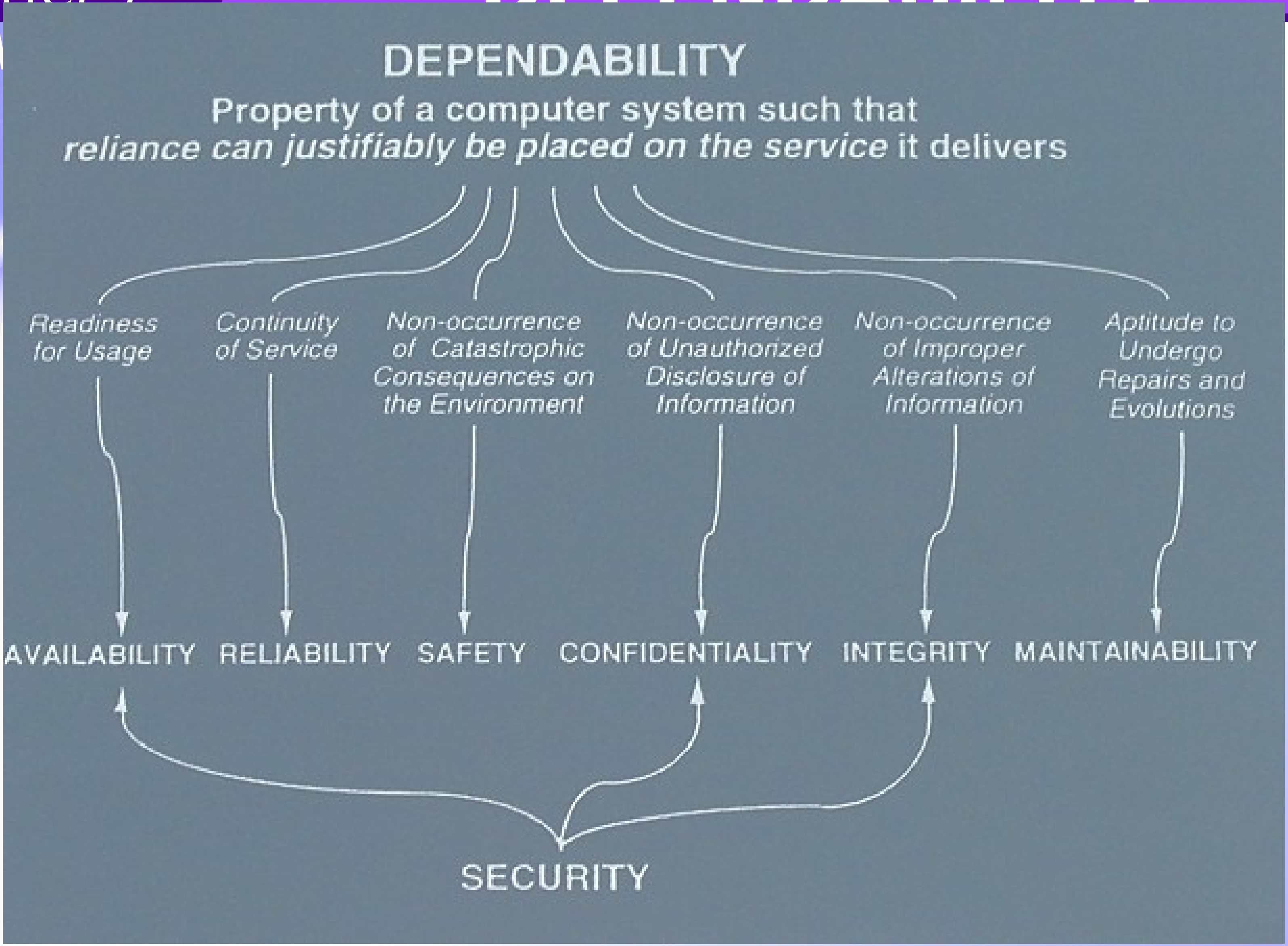
The Attributes of Dependability

➢Dependability is an integrative concept that encompasses the following basic attributes:

- availability: readiness for correct service
- reliability: continuity of correct service
- safety: absence of catastrophic consequences
- confidentiality: absence of unauthorized disclosure of information
- integrity: absence of improper system state alterations
- maintainability: ability to undergo repairs and modifications

➢Depending on the application(s), different emphasis may be put on different attributes.

THE ATTRIBUTES OF DEPENDABILITY


DEPENDABILITY
Property of a computer system such that
reliance can justifiably be placed on the service it delivers
Readiness for Usage
Continuity of Service
Non-occurrence of Catastrophic Consequences on the Environment
Non-occurrence of Unauthorized Disclosure of Information
Non-occurrence of Improper Alterations of Information
Aptitude to Undergo Repairs and Evolutions
AVAILABILITY
RELIABILITY
SAFETY
CONFIDENTIALITY
INTEGRITY
MAINTAINABILITY
SECURITY

Dependability Measures

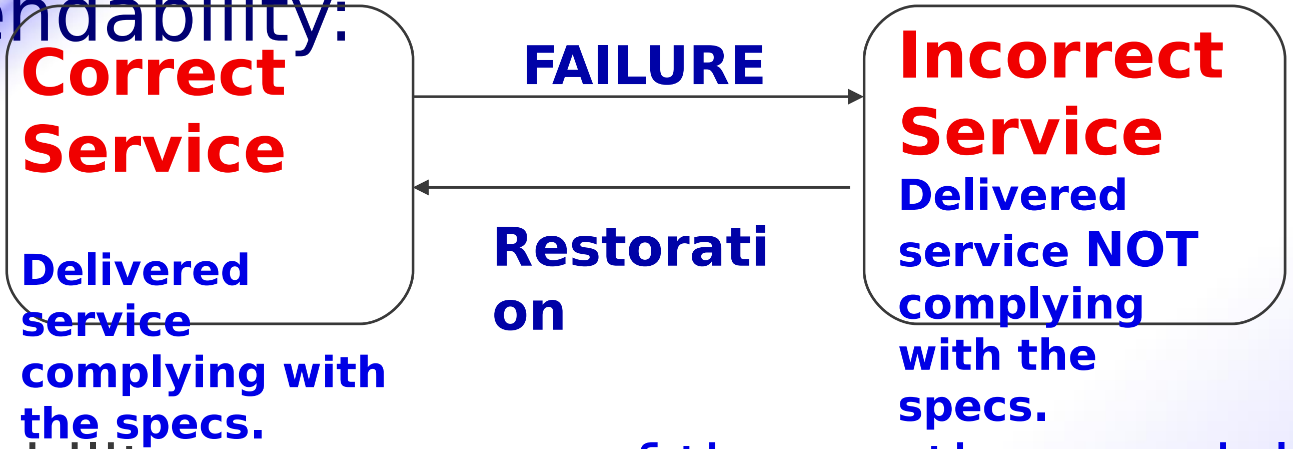
- The dependability attributes of a system should be interpreted in a probabilistic sense: due to the unavoidable presence of faults, systems are never totally available, reliable, safe.....

- The different dependability properties can be measured in terms of probabilities:
- **Reliability R(t):** probability that the system which was working at time 0 is still working at time t, without any failure having occurred *under specified conditions of use*
- **Availability A(t):** probability that the system is "available" (able to deliver service) at time *t*

(difference: repairs affect availability not reliability)

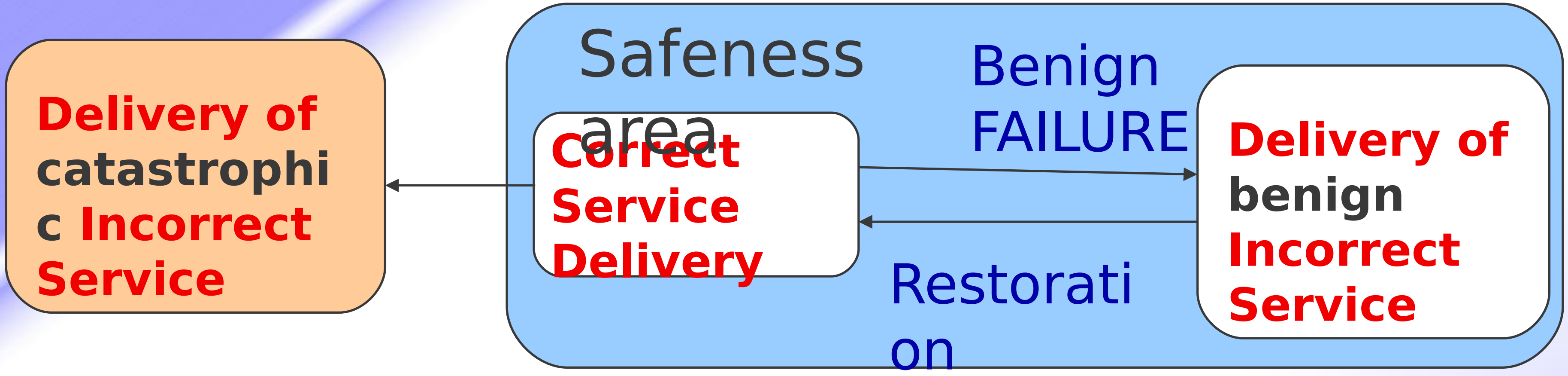
Dependability Measures (cont.d)

➢The alternation of correct-incorrect service delivery is quantified to define the measures of dependability:


Correct Service
Delivered service complying with the specs.
FAILURE
Restoration
Incorrect Service
Delivered service NOT complying with the specs.

- reliability: a measure of the continuous delivery of correct service — or the time to failure,
- availability: a measure of the delivery of correct service with respect to the alternation of correct and incorrect service,
- maintainability: a measure of the time to service restoration since the last failure occurrence,

Safety and Reliability


Delivery of catastrophic Incorrect Service
Safeness area
Correct Service Delivery
Benign FAILURE
Restoration
Delivery of benign Incorrect Service

- safety is an extension of reliability: the state of correct service and the states of incorrect service due to non-catastrophic failure are grouped into a safe state,
- safety is a measure of continuous safeness, or equivalently, of the time to catastrophic failure;
- safety is thus reliability with respect to catastrophic failures.

Safety

- "freedom from dangerous failures"

- Safety = Reliability w.r.t. a certain class of failure
- E.g., traffic light stuck on GREEN: dangerous failure
  stuck on RED: safe failure

- Safety characterises a whole *system*: the traffic light and cars, not just the microcontroller inside the traffic light.

- Safety ≠ reliability or availability !
- They are often conflicting goals

Steady-state availability

- An important measure for repairable systems
- It depends on the frequency of failures, but also on the speed of repairs
- in terms of
  - *mean time to repair*, MTTR and
  - *mean time between failures*, MTBF,
-
- steady-state availability = MTTF / (MTTF + MTTR)

More Specific Measures

- A system may deliver various services or levels of service
  from full capacity to emergency service - *degradable systems*
- and have different failure modes.

- E.g.: telephone switch:
  - may lose half its lines,
  - may lose every other call,
  - may carry all calls with low quality

- Performability : measures of the value of service provided, taking into account failures (combined *performance and dependability* measure)

Forms of dependability requirements

- **Availability :**
  - "the database must be accessible 99% of the time"
- **Rate of occurrence of failures:**
  - "the probability that a failure of a flight control system will cause an accident with fatalities or loss of aircraft must be less than 10^{-9} per hour of flight"
- **Probability of surviving mission:**
  - The probability that the flight and ordnance control system in a fighter plane are still operational at the end of a two hour mission must be more than...

Forms of dependability requirements (cont.d)

- More refined requirements : by class of service, failure mode
  - the database must be accessible 99% of the time for queries, 90 % for updates.
  - the probability that any one terminal cannot access the database for one hour or more in any one day must be less than 1%
- Other forms of requirements
  - Fault tolerance:
    - this system must provide uninterrupted service with up to one component failure, and fail safely if two fail
  - Specific defensive mechanisms:
    - "these data shall be held in duplicate on two disks..

Deriving Dependability Requirements

- Consider role of computing system as a subsystem within a larger system (vehicle, industrial plant, enterprise, ..)
- Analyse potential effects on the larger system of failures of the computer system's services:
  - loss of production
  - damage to plant or environment, goods, life and limb
- Establish acceptable bound on adverse effects
- Quantify adverse effects as a function of the dependability of the computer system
- Establish requirements on computer system dependability so that adverse effects of failures are below the acceptable bound

More on dependability attributes

- Dependability attributes may be emphasized to a different extent depending on the application: always availability (more or less) while reliability, safety, confidentiality may or may not be required.
- This variation in emphasis directly affects the appropriate balance of the techniques to be employed for making the resulting system dependable.
- Some of the attributes are in conflict (e.g. availability and safety, availability and security), necessitating design trade-offs.
- New buzzwords for dependability: resiliency, survivability and trustworthiness.

The Means to Attain Dependability

➢ The development of a dependable computing system calls for the combined utilization of a set of four techniques:

- fault prevention: how to prevent the occurrence or introduction of faults,
- fault tolerance: how to deliver correct service in the presence of faults,
- fault removal: how to reduce the number or severity of faults,
- fault forecasting: how to estimate the present number, the future incidence, and the likely consequences of faults.

Fault Prevention -1

- Fault prevention is attained by *quality control techniques* employed during the design and manufacturing of hardware and software.
- They include *structured programming, information hiding, modularization, etc*., for software, and *rigorous design rules and selection of high-quality, mass-manufactured hardware components* for hardware.
- *Simple design*, possibly at the cost of constraining functionality or increasing cost

Fault Prevention -2

- *Formal proof* of important properties of the design
- Provision of *appropriate operating environment* (air conditioning, protection against mechanical damage) intend to prevent operational physical faults, while *training, rigorous procedures for maintenance, 'foolproof' packages*, intend to prevent interaction faults.
- *Firewalls and similar defenses* intend to prevent malicious faults.

Fault Tolerance

- Fault tolerance is intended to preserve the delivery of correct service in the presence of active faults.
- It is generally implemented by
- **error detection and**
- **subsequent system recovery**.

Error Detection

- **Error detection** originates an error signal or message within the system. An error that is present but not detected is a latent error.
- There exist two classes of error detection techniques:
  - concurrent error detection, which takes place during service delivery,
  - preemptive error detection, which takes place while service delivery is suspended; it checks the system for latent errors and dormant faults. *(preventive maintenance)*

System Recovery

- System Recovery transforms a system state that contains one or more errors and (possibly) faults into a state without detected errors and faults that can be activated again.
- Recovery consists of error handling and fault handling (Fault treatment).

Error Handling

- Error handling eliminates errors from the system state. It may take three forms:
  - Rollback: the state transformation consists of returning the system back to a saved state that existed prior to error detection; that saved state is a checkpoint,
  - Compensation: the erroneous state contains enough redundancy to enable error elimination,
  - Rollforward: the state without detected errors is a new state.

Fault handling (Fault treatment)-1

- Fault handling prevents located faults from being activated again.
- Fault handling involves four steps:
    - fault diagnosis: identifies and records the cause(s) of error(s), in terms of both location and type,
    - fault isolation: performs physical or logical exclusion of the faulty components from further participation in service delivery, i.e., it makes the fault dormant,

Fault handling (Fault treatment) -2

- system reconfiguration: either switches in spare components or reassigns tasks among non-failed components,
- system reinitialization: checks, updates and records the new configuration and updates system tables and records.

- Usually, fault handling is followed by corrective maintenance that removes faults isolated by fault handling.
- Fault tolerance differs from maintenance since maintenance requires the participation of an external agent.

Remarks

- ➢The choice of fault tolerance techniques (using redundancy), and of their implementation, is directly related to the underlying fault assumption.
- ➢For tolerating operational physical faults redundancy may be of identical design, based on the assumption that hardware fail independently.
- ➢For tolerating solid design faults redundancy needs to implement the same function via separate designs and implementations, through design diversity.

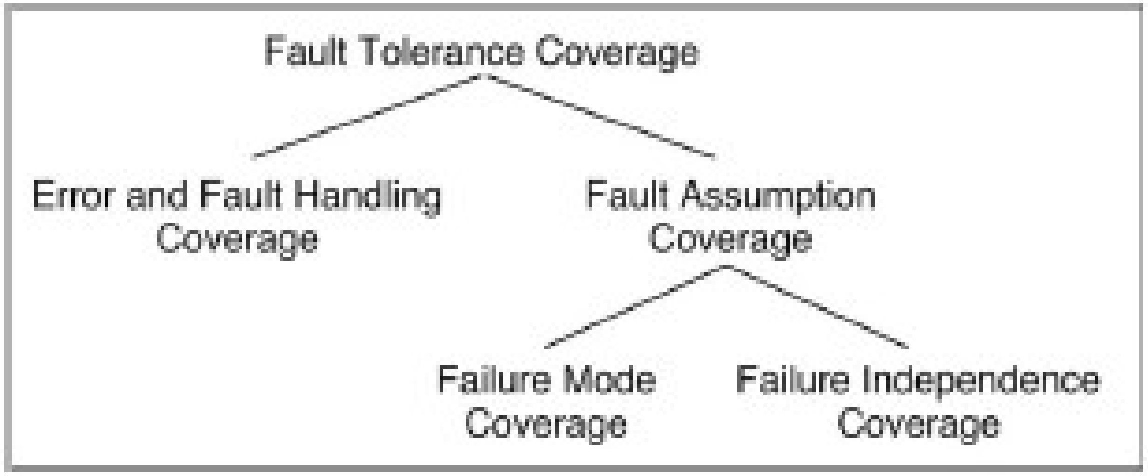
Recursive Fault Tolerance

- Fault tolerance is a recursive concept: the mechanisms that implement fault tolerance should be protected against the faults that might affect them. (voter replication, self-checking checkers, ‘stable’ memory).
- Fault tolerance is not restricted to accidental faults but may also deal with malicious ones: intrusions malicious logic, e.g. viruses.
- Not all fault tolerance techniques are equally effective. The measure of effectiveness of any given FT technique is called its coverage.

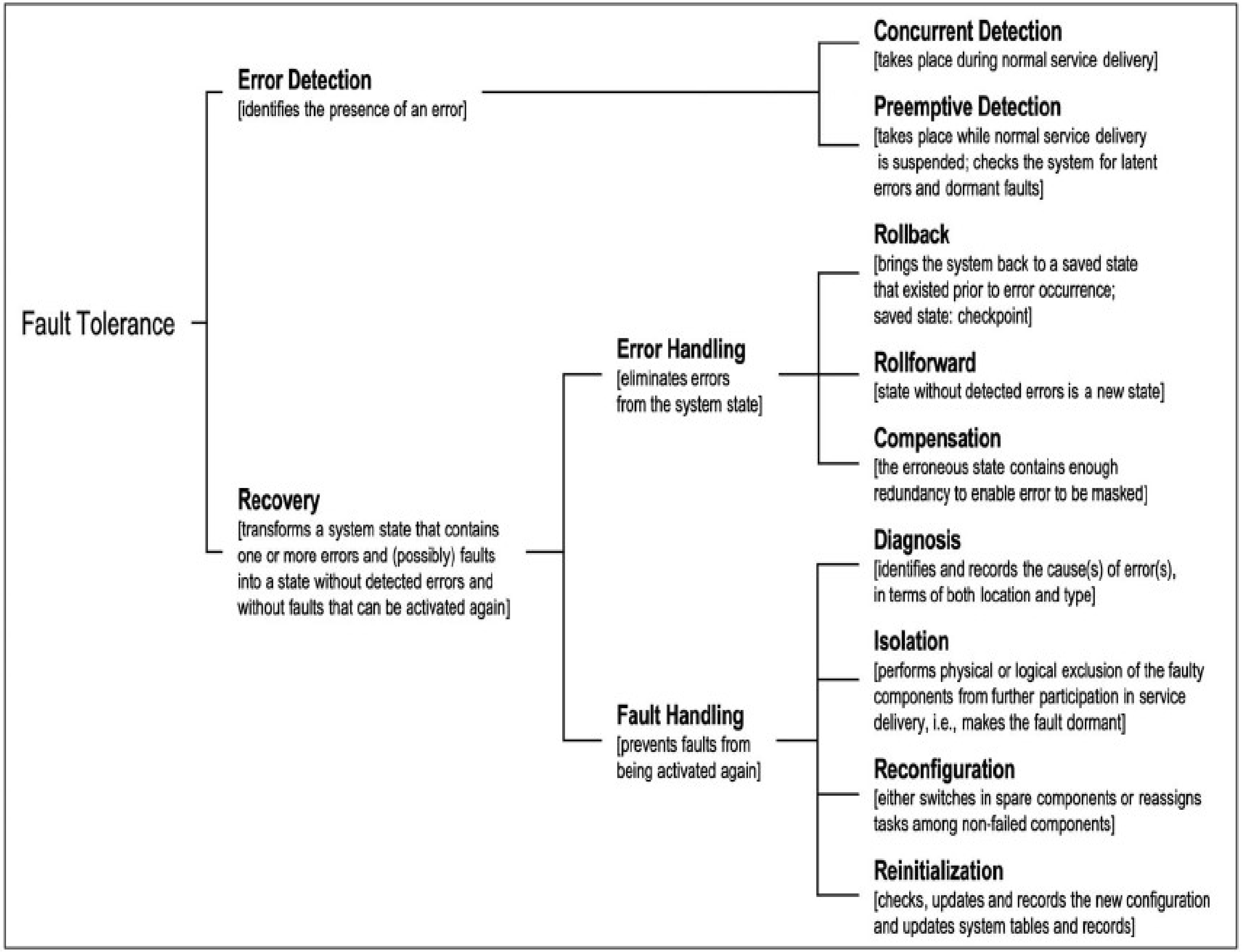
Fault Tolerance coverage

- The lack of fault tolerance coverage is a severe limitation to the increase in dependability that can be obtained.
- Such imperfections of fault tolerance are due either
  - to development faults of the fault tolerance mechanisms with respect to the fault assumptions stated, the consequence of which is a lack of error and fault handling coverage
  - to fault assumptions that differ from the faults really occurring in operation, resulting in a lack of fault assumption coverage, that can be in turn due to either
    - failed component(s) not behaving as assumed, that is a lack of failure mode coverage,
    - the occurrence of common-mode failures when independent ones are assumed, that is a lack of failure independence coverage.

More on Fault Tolerance - 4


Fault Tolerance Coverage
Error and Fault Handling Coverage
Fault Assumption Coverage
Failure Mode Coverage
Failure Independence Coverage

Structure of the Fault Tolerance Coverage

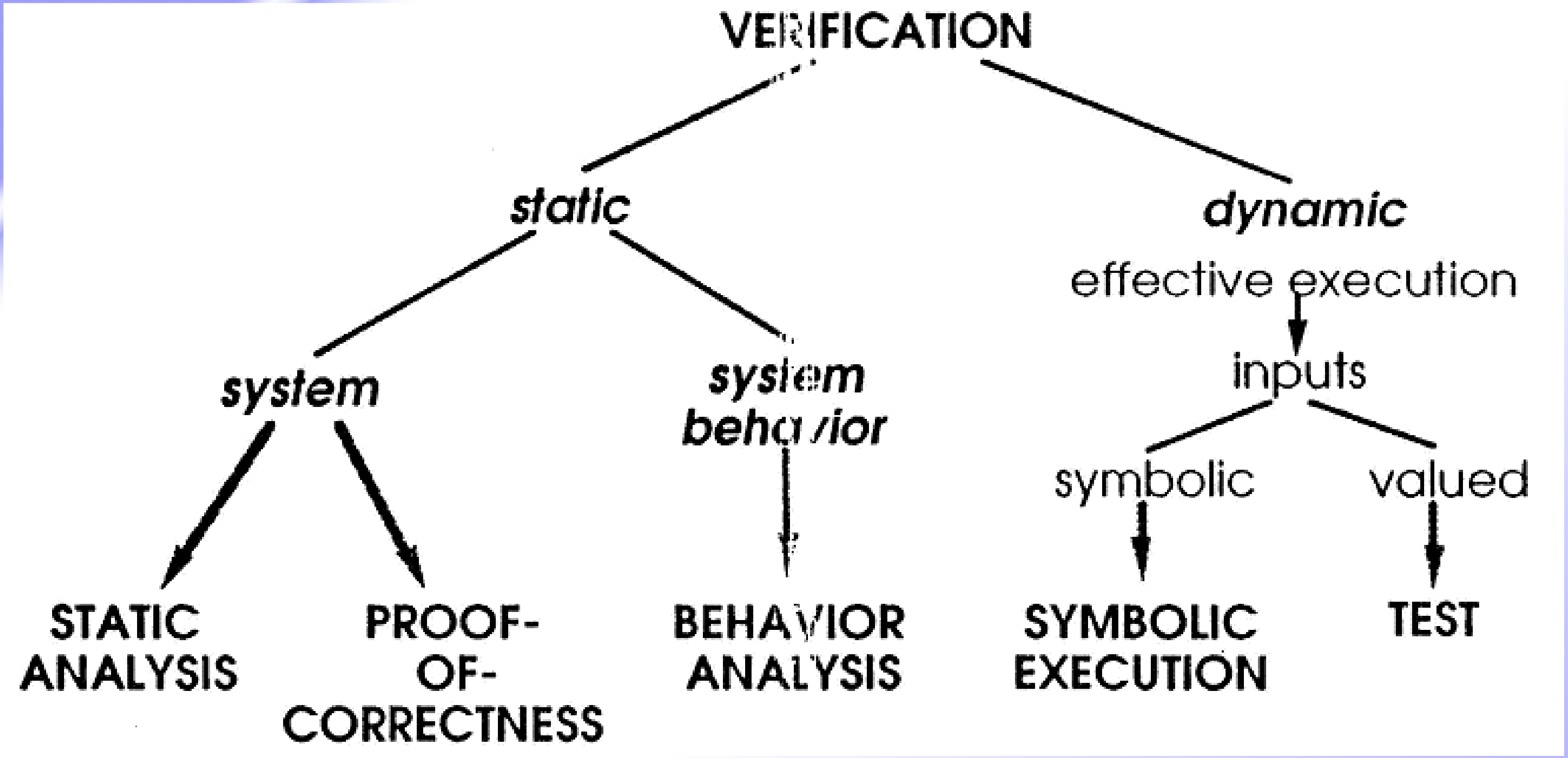


- Fault Tolerance
  - **Error Detection** [identifies the presence of an error]
    - **Concurrent Detection** [takes place during normal service delivery]
    - **Preemptive Detection** [takes place while normal service delivery is suspended; checks the system for latent errors and dormant faults]
  - **Recovery** [transforms a system state that contains one or more errors and (possibly) faults into a state without detected errors and without faults that can be activated again]
    - **Error Handling** [eliminates errors from the system state]
      - **Rollback** [brings the system back to a saved state that existed prior to error occurrence; saved state: checkpoint]
      - **Rollforward** [state without detected errors is a new state]
      - **Compensation** [the erroneous state contains enough redundancy to enable error to be masked]
    - **Fault Handling** [prevents faults from being activated again]
      - **Diagnosis** [identifies and records the cause(s) of error(s), in terms of both location and type]
      - **Isolation** [performs physical or logical exclusion of the faulty components from further participation in service delivery, i.e., makes the fault dormant]
      - **Reconfiguration** [either switches in spare components or reassigns tasks among non-failed components]
      - **Reinitialization** [checks, updates and records the new configuration and updates system tables and records]

Fault Removal

- Fault removal is performed both during the development, and during the operational life of a system.
- During development it consists of three steps: verification, diagnosis, correction.
- Verification is the process of checking whether the system adheres to given properties. If it does not, the other two steps follow.
- After correction, verification should be repeated to check that fault removal had no undesired consequences; the verification performed at this stage is usually termed non-regression verification.

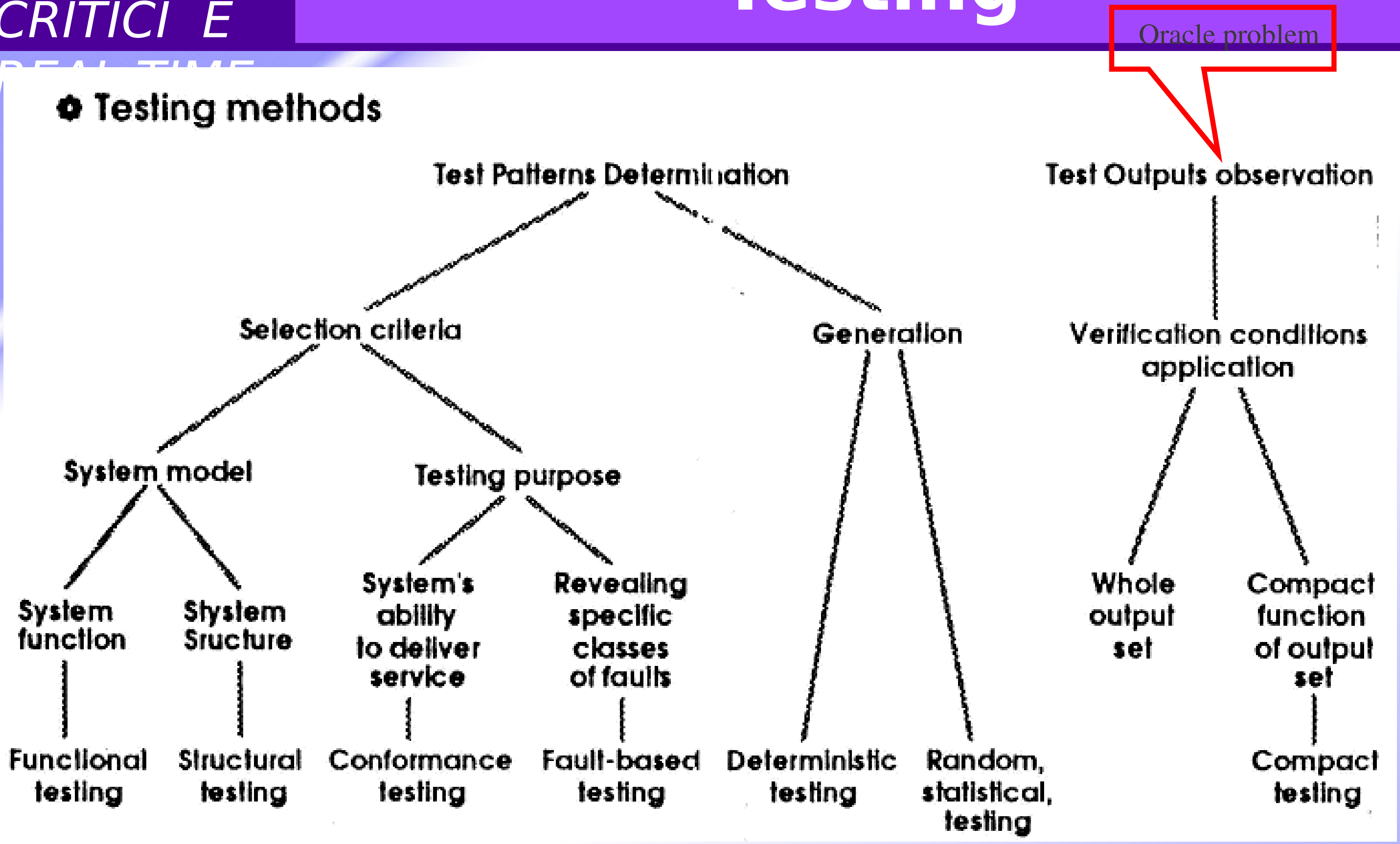
Verification



Verification during development

- ➢Verification techniques can be classified according to whether or not they exercise the system.
- ➢Without actual execution is static verification: static analysis (e.g., inspections or walk-through), model-checking, theorem proving.
- ➢Exercising the system is dynamic verification: either with symbolic inputs in the case of symbolic execution, or actual inputs in the case of testing.
- ➢Important is the verification of fault tolerance mechanisms, especially a) formal static verification, and b) testing that includes faults or errors in the test patterns: fault injection.
- ➢As well as verifying that the system cannot do more than what is specified important to safety and security.

Testing

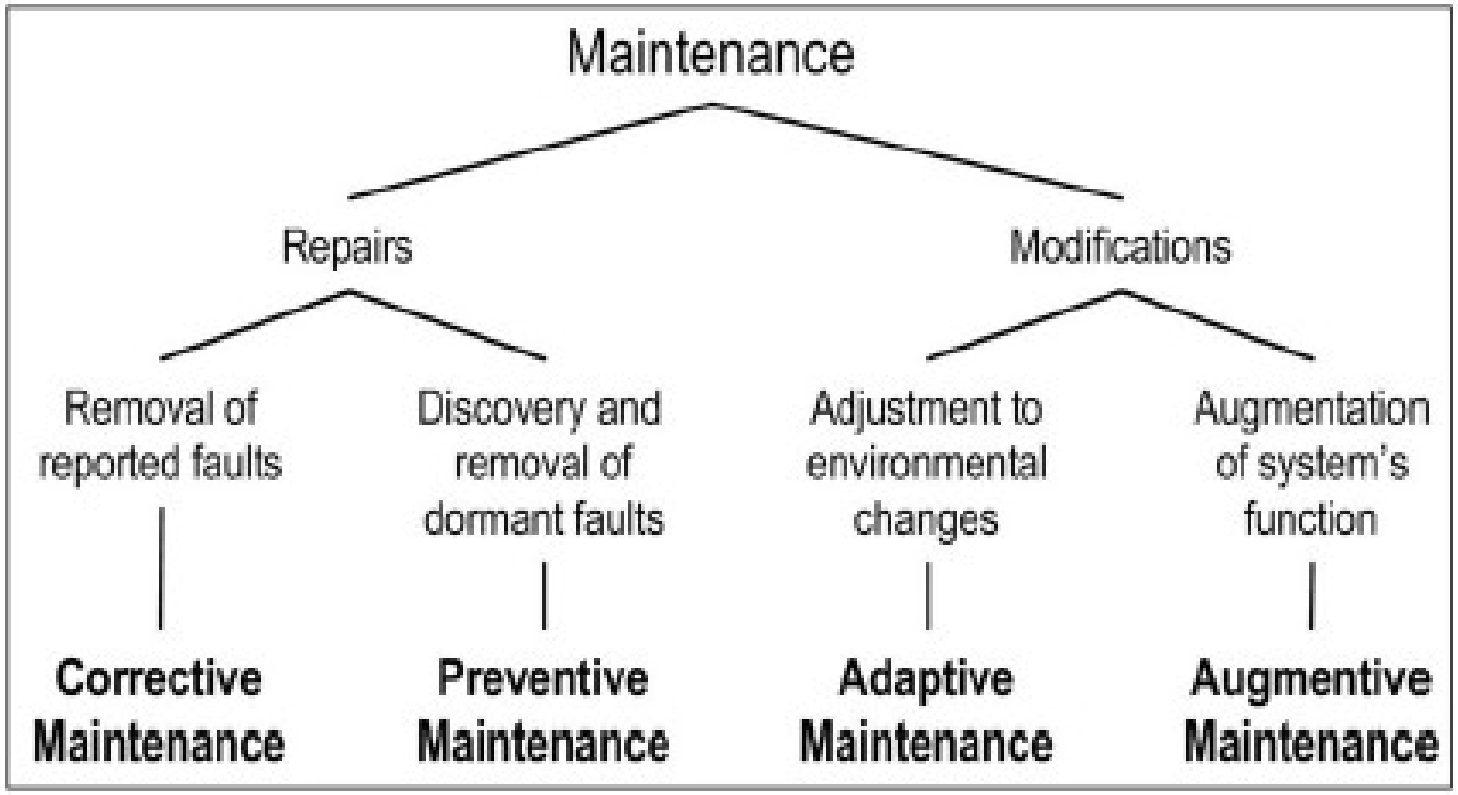

Oracle problem
Testing methods
Test Patterns Determination
Test Outputs observation
Selection criteria
Generation
Verification conditions application
System model
Testing purpose
System function
Stystem Sructure
System's ability to deliver service
Revealing specific classes of faults
Whole output set
Compact function of output set
Functional testing
Structural testing
Conformance testing
Fault-based testing
Deterministic testing
Random, statistical, testing
Compact testing

Verification during the operational life

- Fault removal during the operational life of a system is corrective or preventive maintenance.
- Corrective maintenance is aimed at removing faults that have produced one or more errors and have been reported,
- Preventive maintenance is aimed to uncover and remove faults before they might cause errors during normal operation. a) physical faults that have occurred since the last preventive maintenance actions, and b) design faults that have led to errors in other similar systems.
- These forms of maintenance apply to non-fault-tolerant systems as well as fault-tolerant systems, that can be maintainable on-line (without interrupting service delivery) or off-line (during service outage).

Forms of Maintenance

- The various forms of 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

Maintenance, Dependability and cost

Dependability ⟺ Maintenance

- Systems designed to be dependable **NEED** to undergo maintenance to keep the means for dependability, the systems are equipped with, effective along time

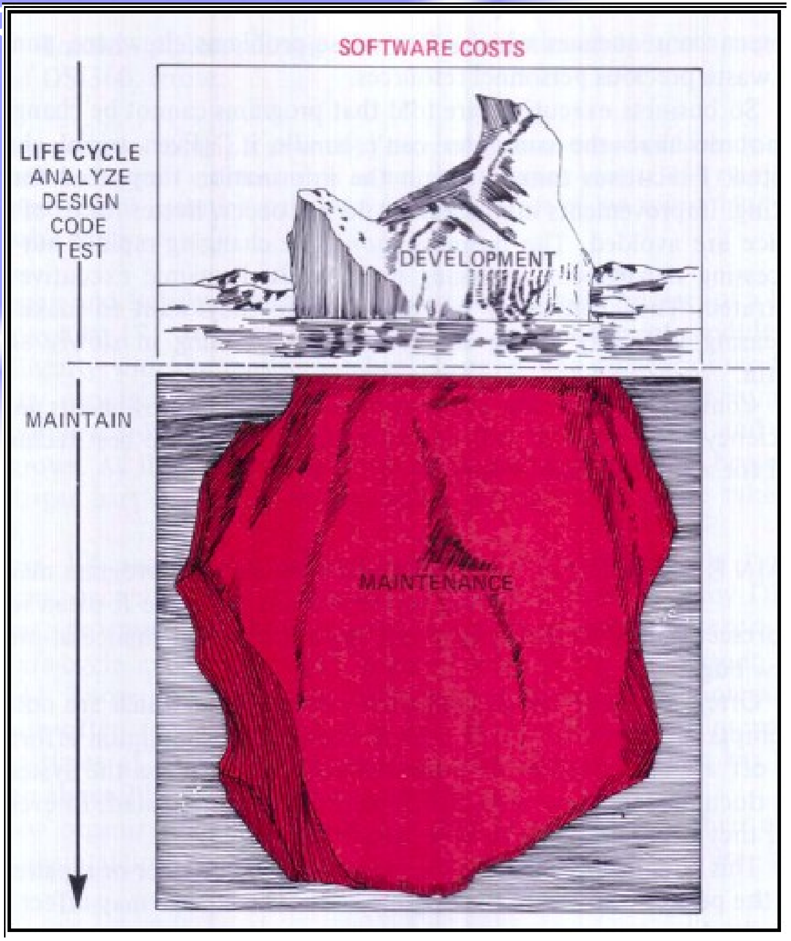
- At the same time, a poorly designed system cannot significantly improve its dependability/QoS just relying on maintenance policies.

Cost of Maintenance

➢ Influenced by several factors, including:

- Decisions on the design (e.g., resorting to components/engineering practices which favour maintainability),
- The approach to the maintenance (e.g., repair or replacement),
- Who is in charge of maintenance (the owner, the manufacturer, or a third party),
- Maintenance contracts

Cost of Maintenance



- Software Maintenance dominates the software life cycle in terms of effort and cost.
- In many organizations, software maintenance activities consume three/fourth of the total life-cycle expenditures.

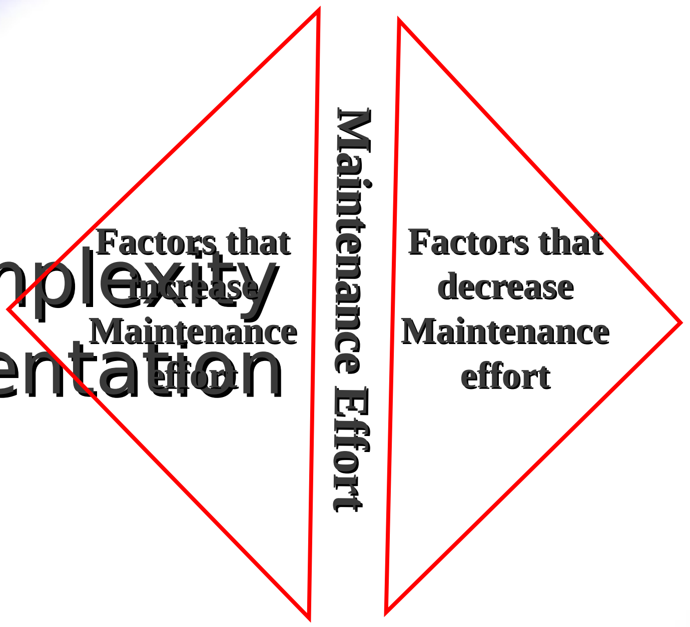
Figure taken from "Software Maintenance", J. Martin, C. McLure, Prentice Hall, 1983

50% to 75% of overall costs- p.533 Software Engineering Economics, B. Boehm

67% p. 24 *Software Maintenance*, J. Martins & C. McClure

Maintenance Effort

- System age
- System size
- Program complexity
- Poor documentation
- ……….


Factors that increase Maintenance effort
Maintenance Effort
Factors that decrease Maintenance effort

- Use of structured techniques
- Use of automated tools
- “Team programming concept”
- Experience of maintainers
- Clear and unambiguous documentation
- ……

The most effective way to deal with maintenance is to apply the “design for maintainability” principle since the early stages of the system development.

The role of maintenance and repair

- High dependability may be obtained by combinations of
  - making faults very unlikely
  - making faults very likely to be tolerated
  - repairing faults promptly
- Fault avoidance, fault tolerance may substitute for manual repair/maintenance where this is unfeasible - at a cost
- More commonly, fault tolerance techniques allow:
  - continued acceptable service pending repair, “graceful degradation” of service: reduced cost of faults
  - deferred repair, repair with scheduled maintenance: reduced cost of maintenance/repair
  - more effective maintenance/repair

Fault Forecasting

- Fault forecasting is conducted by performing an evaluation of the system behavior with respect to fault occurrence or activation.
- Evaluation has two aspects:
  - qualitative: aims to identify, classify, rank the failure modes, or the event combinations (component failures or environmental conditions) that would lead to system failures
  - quantitative, or probabilistic: which aims to evaluate in terms of probabilities the extent to which the relevant attributes of dependability are satisfied

Fault Forecasting: methods and objectives

- The methods for qualitative and quantitative evaluation are
  - either specific (e.g., FMEA for qualitative evaluation, or Markov chains and stochastic Petri nets for quantitative evaluation)
  - or can be used to perform both forms of evaluation (e.g., reliability block diagrams, fault-trees).

- Failure intensity, the number of failures per unit of time measures the frequency of system failures. Failure intensity typically first decreases (reliability growth), then stabilizes (stable reliability), then increases (reliability decrease).

Probabilistic fault-forecasting

➢The two main approaches to probabilistic fault-forecasting aiming to derive probabilistic estimates are
- modeling (analytical or simulation) and
- (evaluation) testing

➢These approaches are complementary,

➢since modeling needs data on the basic processes (failure process, maintenance process, system activation process, etc.),

➢that may be obtained either by testing, or by the processing of failure data.

➢For fault-tolerant systems, the effectiveness of error and fault handling mechanisms (their coverage) has a drastic influence on dependability measures and can be measured either through modeling or through testing i.e. fault injection.

Model-based analysis

A **Model** is an **abstraction** of a system

“that highlights the important features of the system organization and provides ways of quantifying its properties neglecting all those details that are relevant for the actual implementation, but **that are marginal for the objective of the study**”(*)

Therefore:

- Making the **right assumptions** on system components and environment is a crucial issue
- **Assumptions** depend on the **indicator** under analysis

(*)*Definition from G. Balbo, “Introduction to stochastic petri nets”, Lectures on Formal Methods and Performance Analysis, volume 2090 of Lecture Notes in Computer Science, pages 84-155. Springer Verlag, 2001*

Evaluation Techniques

There are several choices:

❑**Model-based solutions**
- Combinatorial modeling
- Analytic/numerical modeling
- Simulation (including fault injection on a simulated system

❑**Measurement** (including performance benchmarking and fault injection on a prototype system)

Each with differing advantages and disadvantages

Choice

Choice of a method depends on:

- Stage of design (is it a proposed or existing system?)
- Time (how long until results are required)
- Tools available
- Accuracy
- Ability to compare alternatives
- Cost
- Scalability

Choosing an evaluation technique

Criterion	Combinatorial	State-based space	Simulation	Measurement
Stage	Any	Any	Any	Post-prototype
Time	Small	Medium	Medium	Varies
Tools	Formulae, spreadsheets	Languages & Tools	Languages & Tools	Instrumentati on
Accuracy	Low	Moderate	Moderate	High
Comparison	Easy	Moderate	Moderate	Difficult
Cost	Low	Low/Medium	Medium	High
Scalability	High	Low/Medium	Medium	Low

Model Validation

- Model validation is the process of making sure that the **model** you build is **correct**.

Correctness means two things:

1) The model **accurately represents** the system,
2) The **model specified is the model intended**.

- **Models are frequently validated by three methods:**
  - **Measurements**: measures on the model should match measures on the real system,
  - **Theoretical results**:
    - measure something to which you already know the answer, e.g., throughput cannot exceed capacity,
  - **Insight of experts**:
    - Modeler expertise comes with experience,

Fault Injection

- An approach to quantitative assessment of dependability attributes, especially for hardware components
- Typical applications:
  - tracing of error propagation
  - estimation of the error latency
  - estimation of the coverage provided by fault tolerance algorithms and mechanisms
- How it works:
  - Faults are introduced at various levels of the system architecture, to analyse their impact on the system behaviour
  - Several techniques and methods have been developed and integrated in specific tools

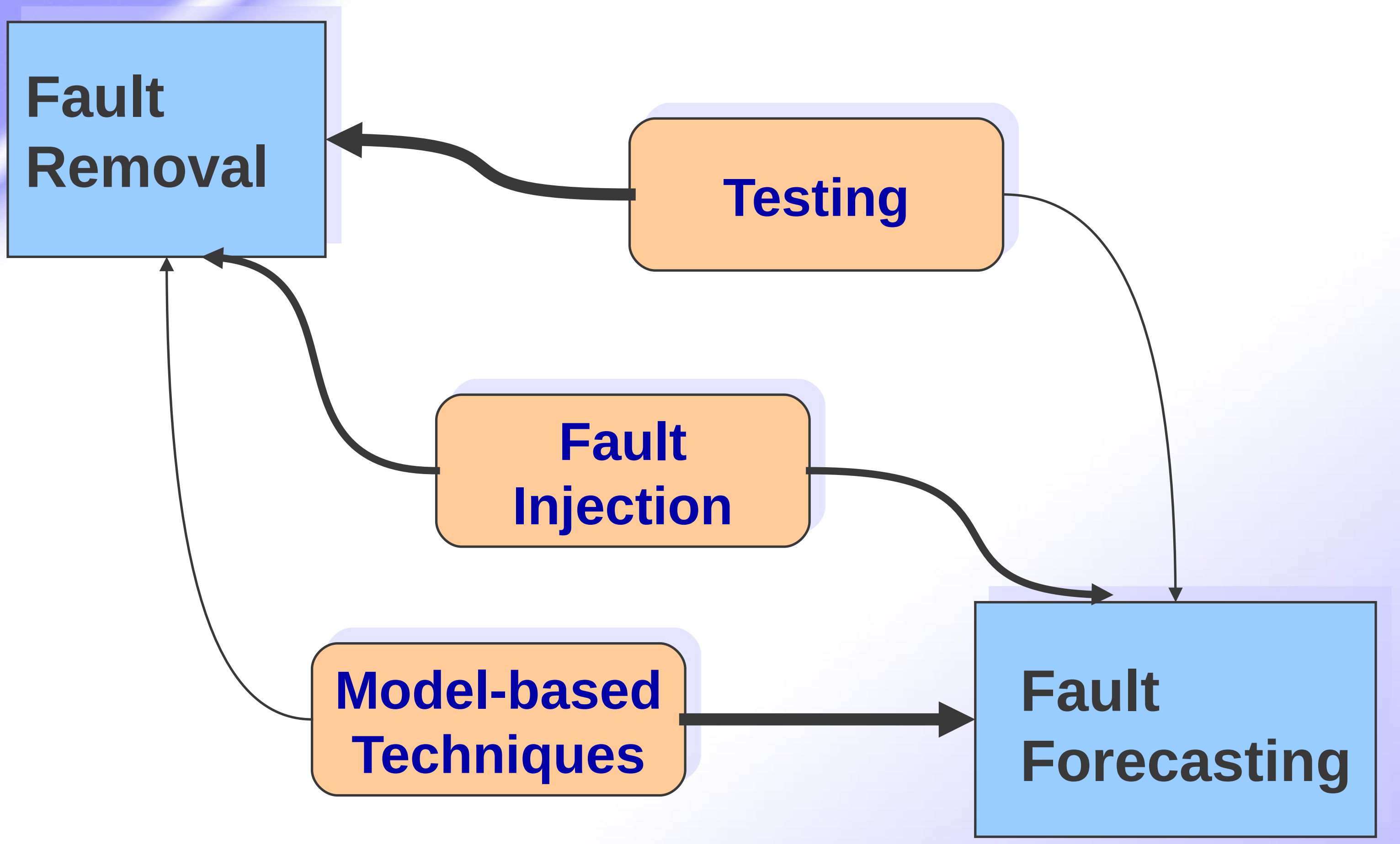
Fault injection provides partial results

- *---> a percentage x% of the injected faults while the system is running component y results in a detected error*

- Statistical techniques are required to derive coverage estimation from the results of a fault injection experiment

- Two objectives:
  - minimize the error for the tested part of the system
    - e.g., taking into account the fact that: i) faults have not the same probability of occurrence;ii) the injected errors may be not representative for the system
    - also through the usage of other fault-forecasting techniques
  - extend the results obtained to the non-tested part
    - The confidence intervals of the coverage estimate for the whole system are usually too wide
    - Information on the system structure can be usefully exploited , e.g., in the case of physical faults injected at pin level of IC

Relations between fault removal and forecasting

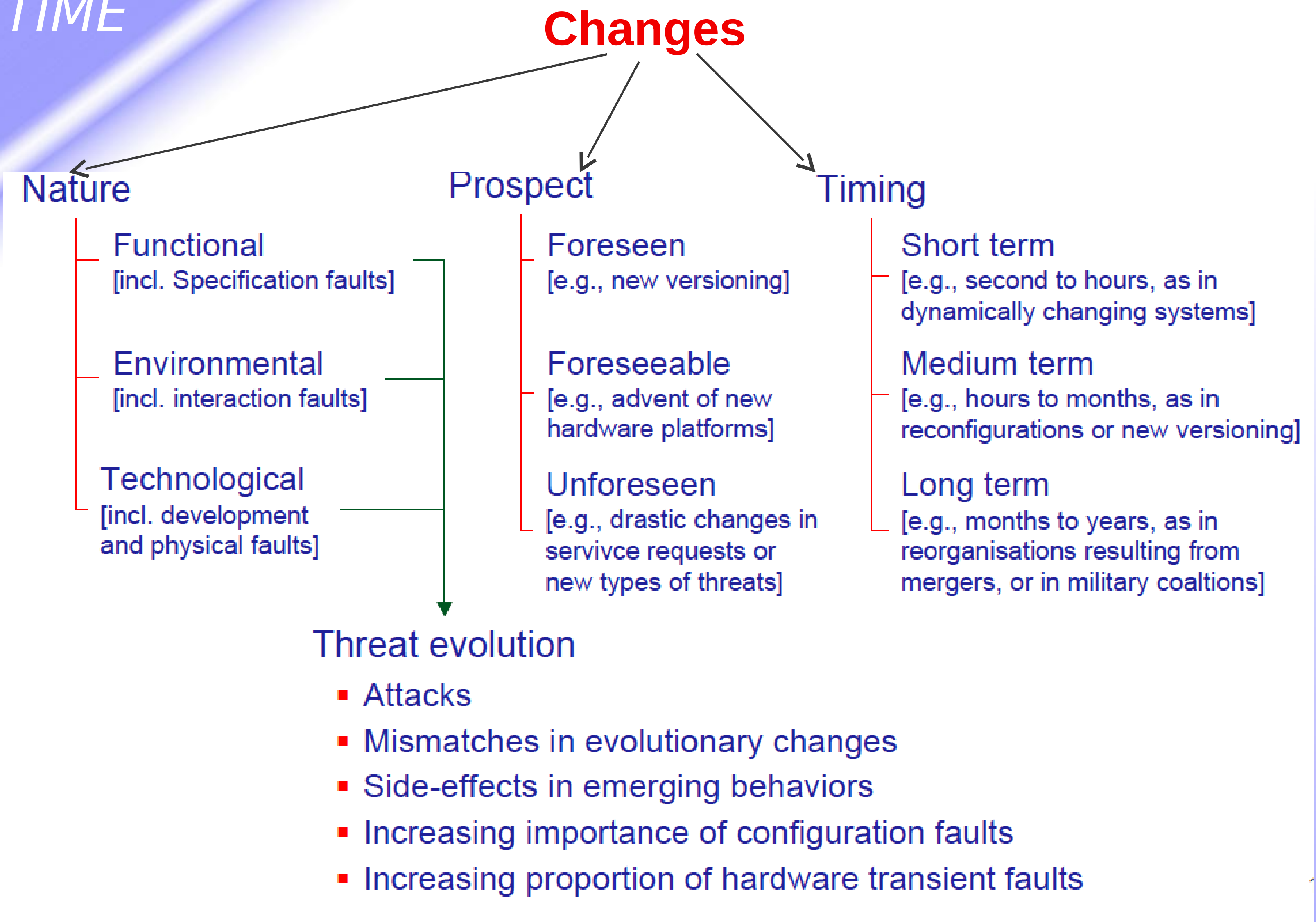


Evolution of the Dependability concept: Resilience

- RESILIENCE: persistence of Dependability when facing changes
- Adjective Resilient in use for 30+ years
- in preface of “Resilient Computing Systems” T. Anderson (Ed.), Collins, 1985:

  “The two key attributes are *dependability* and *robustness* […]. A computing system can be *robust* if it retains its ability to deliver service in conditions that are beyond its normal domain of operation”
- Used as a new buzzword, essentially as synonym or substitute to, fault tolerant
- Generalization of Fault Tolerance: Change Tolerance

Changes classification



Changes

- Nature
  - Functional [incl. Specification faults]
  - Environmental [incl. interaction faults]
  - Technological [incl. development and physical faults]
- Prospect
  - Foreseen [e.g., new versioning]
  - Foreseeable [e.g., advent of new hardware platforms]
  - Unforeseen [e.g., drastic changes in servivce requests or new types of threats]
- Timing
  - Short term [e.g., second to hours, as in dynamically changing systems]
  - Medium term [e.g., hours to months, as in reconfigurations or new versioning]
  - Long term [e.g., months to years, as in reorganisations resulting from mergers, or in military coaltions]

Threat evolution

- Attacks
- Mismatches in evolutionary changes
- Side-effects in emerging behaviors
- Increasing importance of configuration faults
- Increasing proportion of hardware transient faults

Conclusions

- ➢Dependability subsumes concerns in Reliability, Availability, Safety, Confidentiality, Integrity, Maintainability, Performability, ...
- ➢with a unified conceptual framework and enables the appropriate balance of these properties to be addressed.

- ➢Means for dependability (prevention, tolerance removal and forecasting) provide an orthogonal classification for development activities

- ➢Fault-Error-Failure chain is central to understanding impairments likely to affect a system

- ➢Rigorous terminology: avoids confusion.

New buzzwords for, and extension of the concept of, dependability: resiliency, survivability and trustworthiness.