Dependable Computing: Concepts, Challenges, Directions

Jean-Claude Laprie

COMPSAC 2004 — Hong Kong, September 28-30, 2004

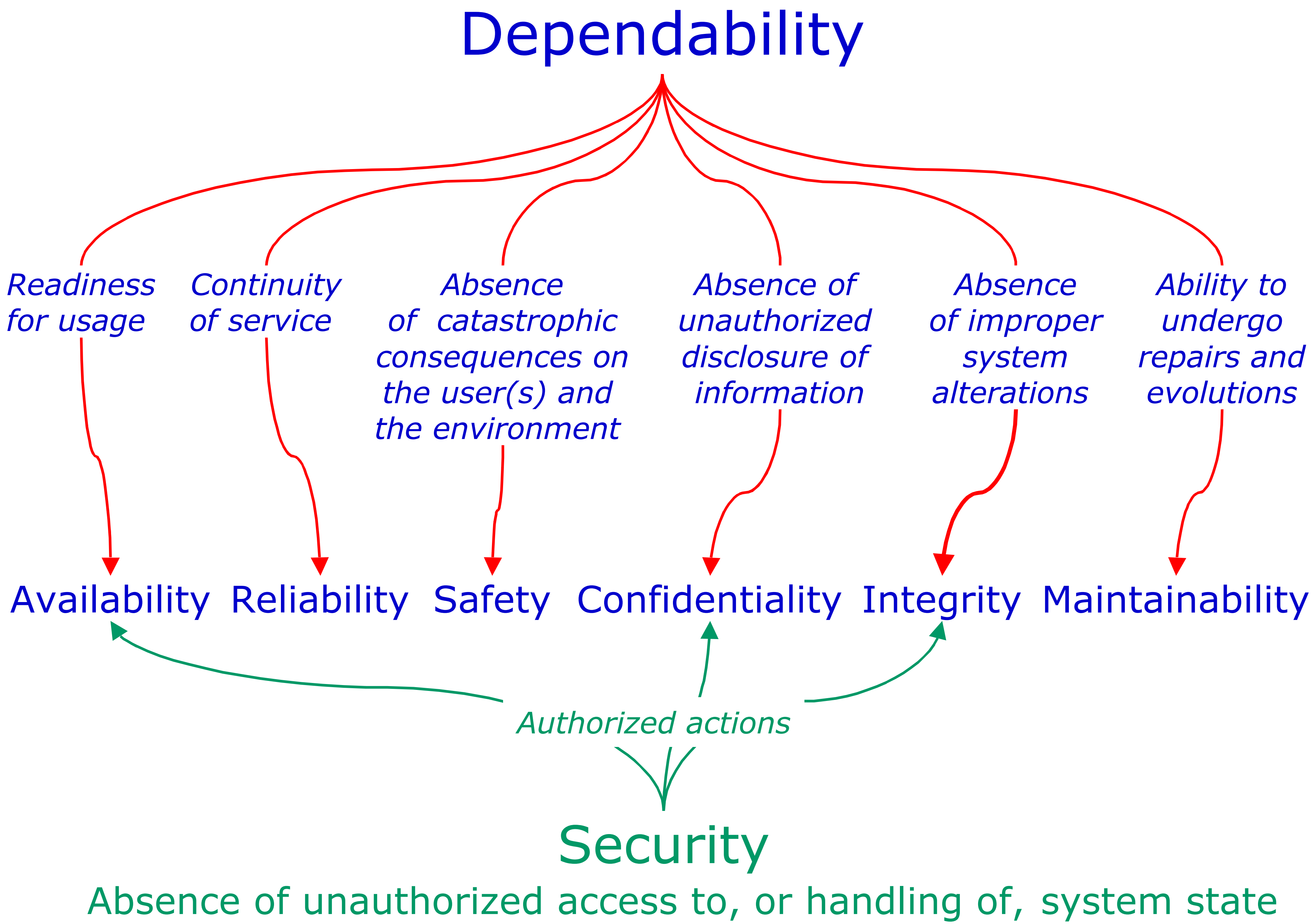
- Concepts
  - A. Avizienis, J.C. Laprie, B. Randell, C. Landwehr: ‘Basic Concepts and Taxonomy of Dependable and Secure Computing’, *IEEE Trans. on Dependable and Secure Computing*, vol. 1, no. 1, Jan-March 2004, pp. 11-33
- Challenges
  - From real-life statistical data
- Directions
  - For ubiquitous computing to be effective

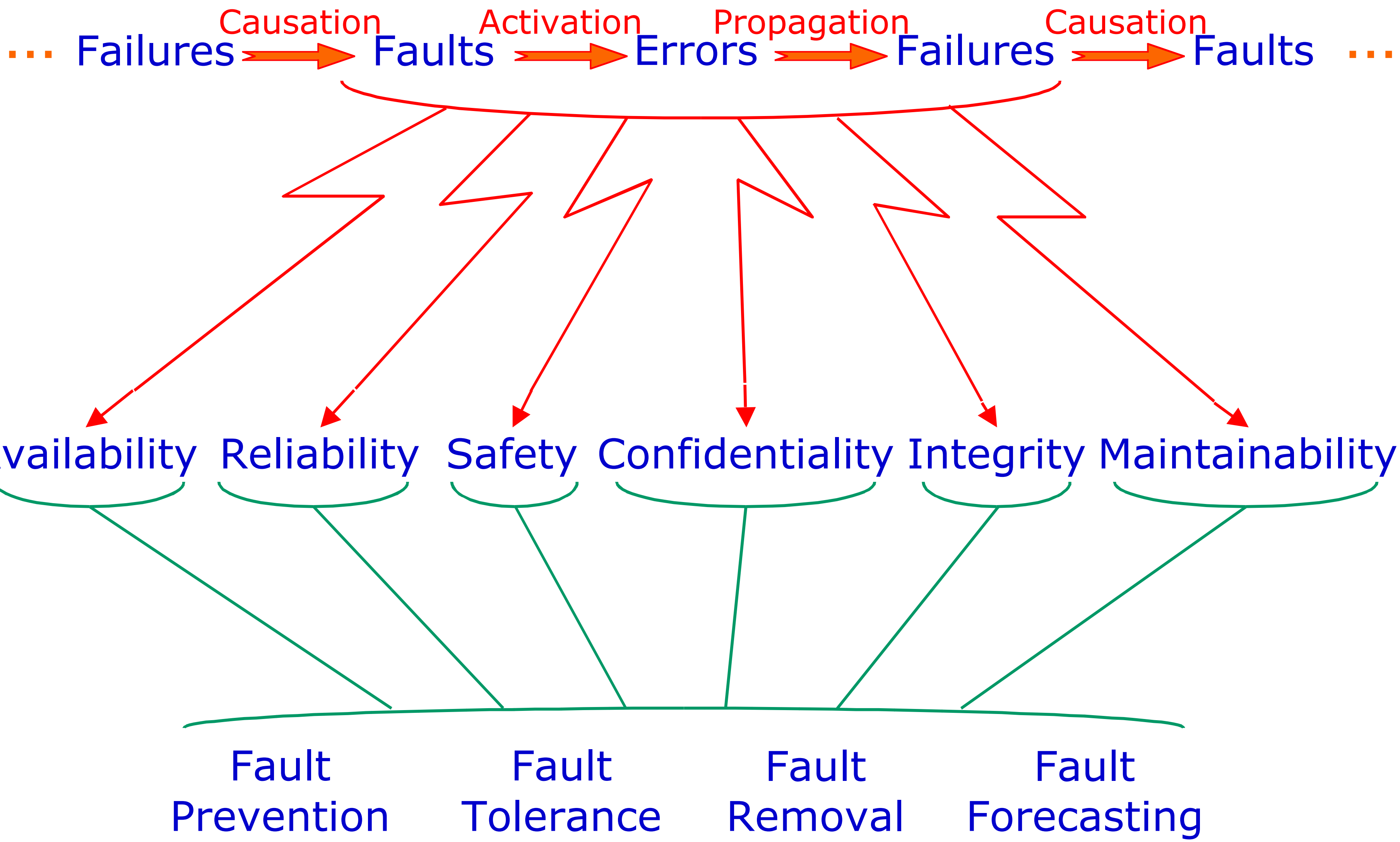
Dependability: ability to deliver service that can justifiably be trusted

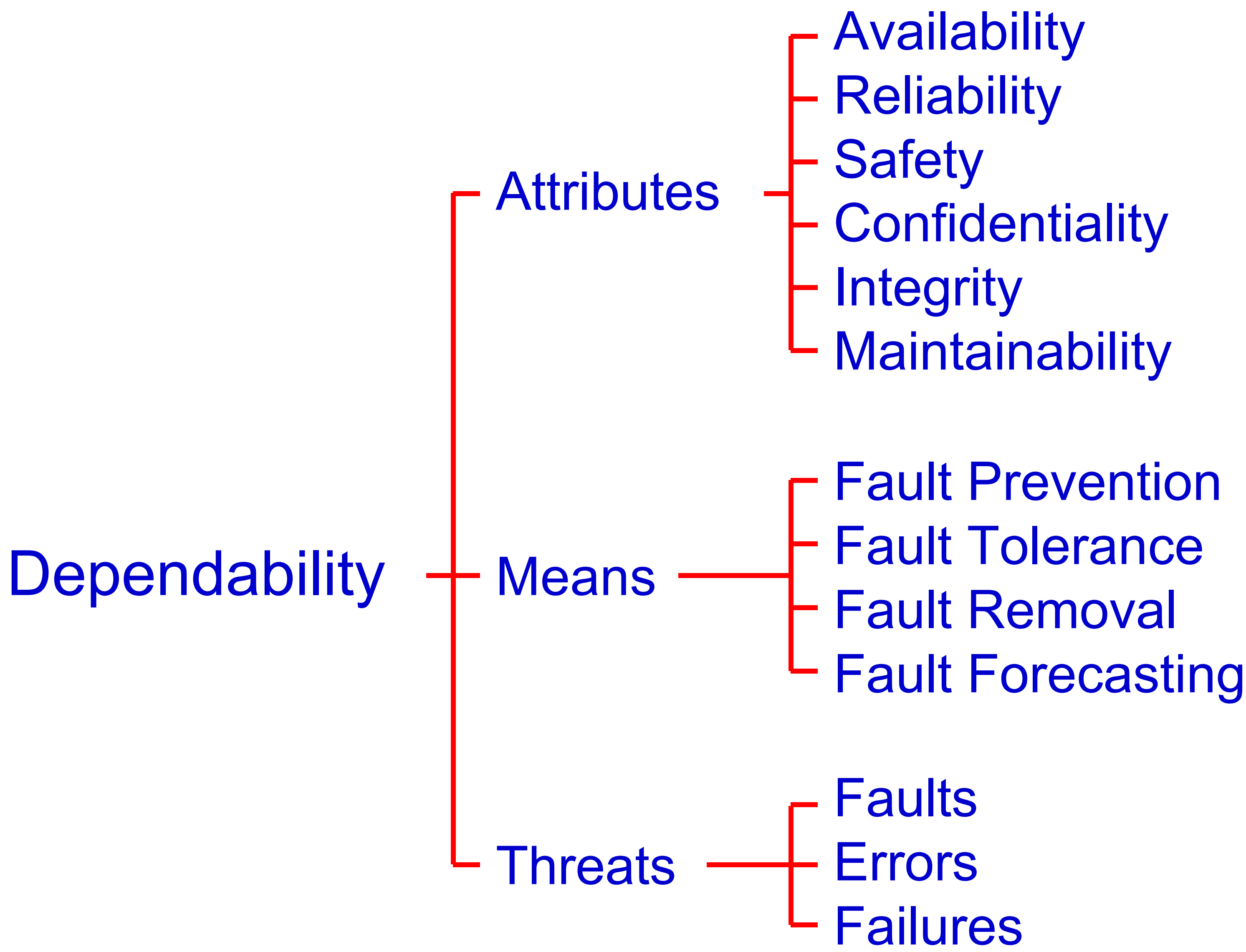
- **Service** delivered by a system: its behavior as it is perceived by its user(s)
- **User**: another system that interacts with the former
- **Function** of a system: what the system is intended to do
- (Functional) **Specification**: description of the system function
- **Correct service**: when the delivered service implements the system function
- **Service failure**: event that occurs when the delivered service deviates from correct service, either because the system does not comply with the specification, or because the specification did not adequately describe its function
- Part of system state that may cause a subsequent service failure: **error**
  Adjudged or hypothesized cause of an error: **fault**
- **Failure modes**: the ways in which a system can fail, ranked according to failure severities

Dependability: ability to avoid service failures that are more frequent or more severe than is acceptable

- When service failures are more frequent or more severe than acceptable: **dependability failure**


Dependability
Readiness for usage
Continuity of service
Absence of catastrophic consequences on the user(s) and the environment
Absence of unauthorized disclosure of information
Absence of improper system alterations
Ability to undergo repairs and evolutions
Availability
Reliability
Safety
Confidentiality
Integrity
Maintainability
Authorized actions
Security
Absence of unauthorized access to, or handling of, system state

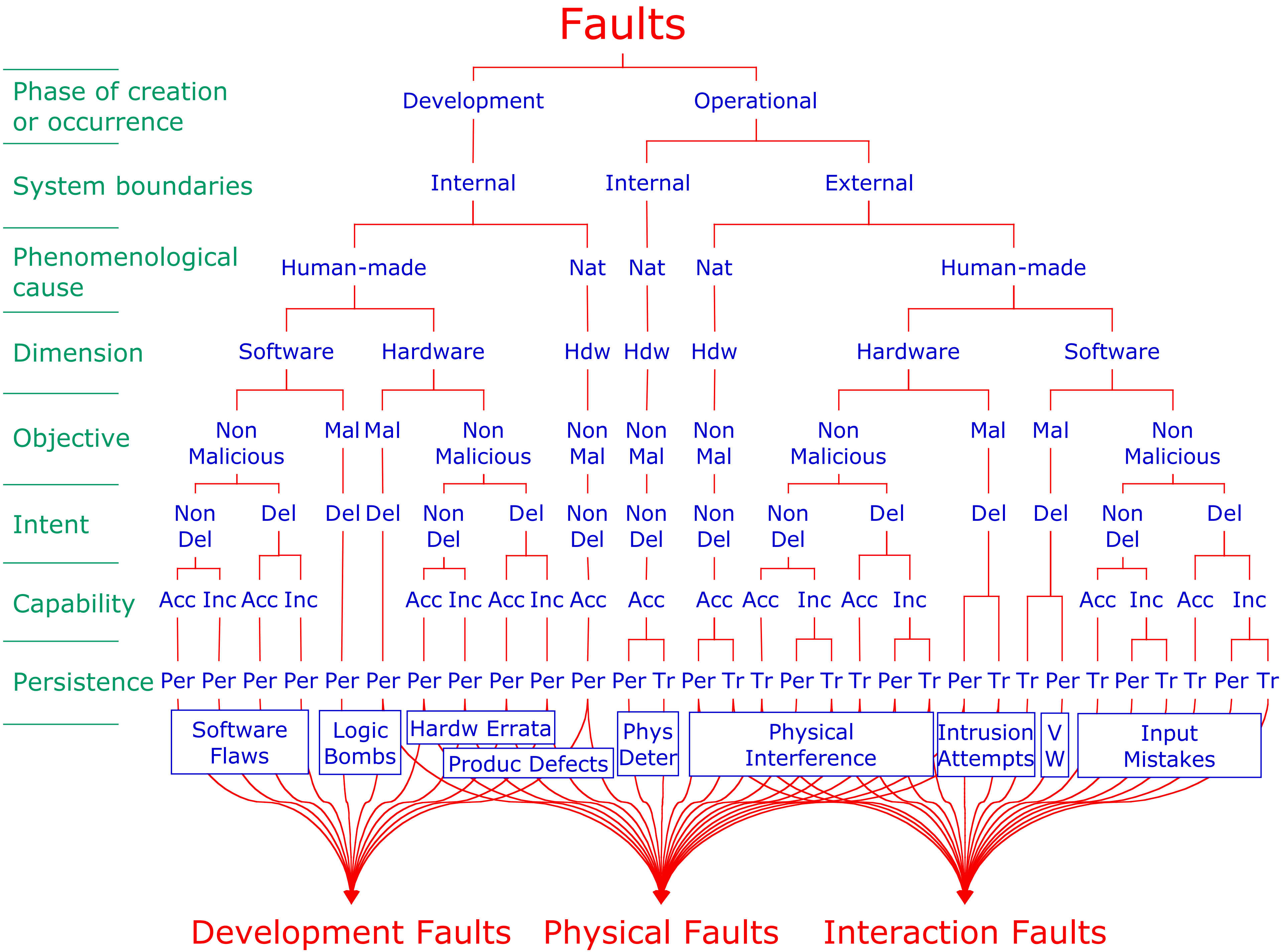

... Failures
Causation
Faults
Activation
Errors
Propagation
Failures
Causation
Faults ...
Availability
Reliability
Safety
Confidentiality
Integrity
Maintainability
Fault Prevention
Fault Tolerance
Fault Removal
Fault Forecasting


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

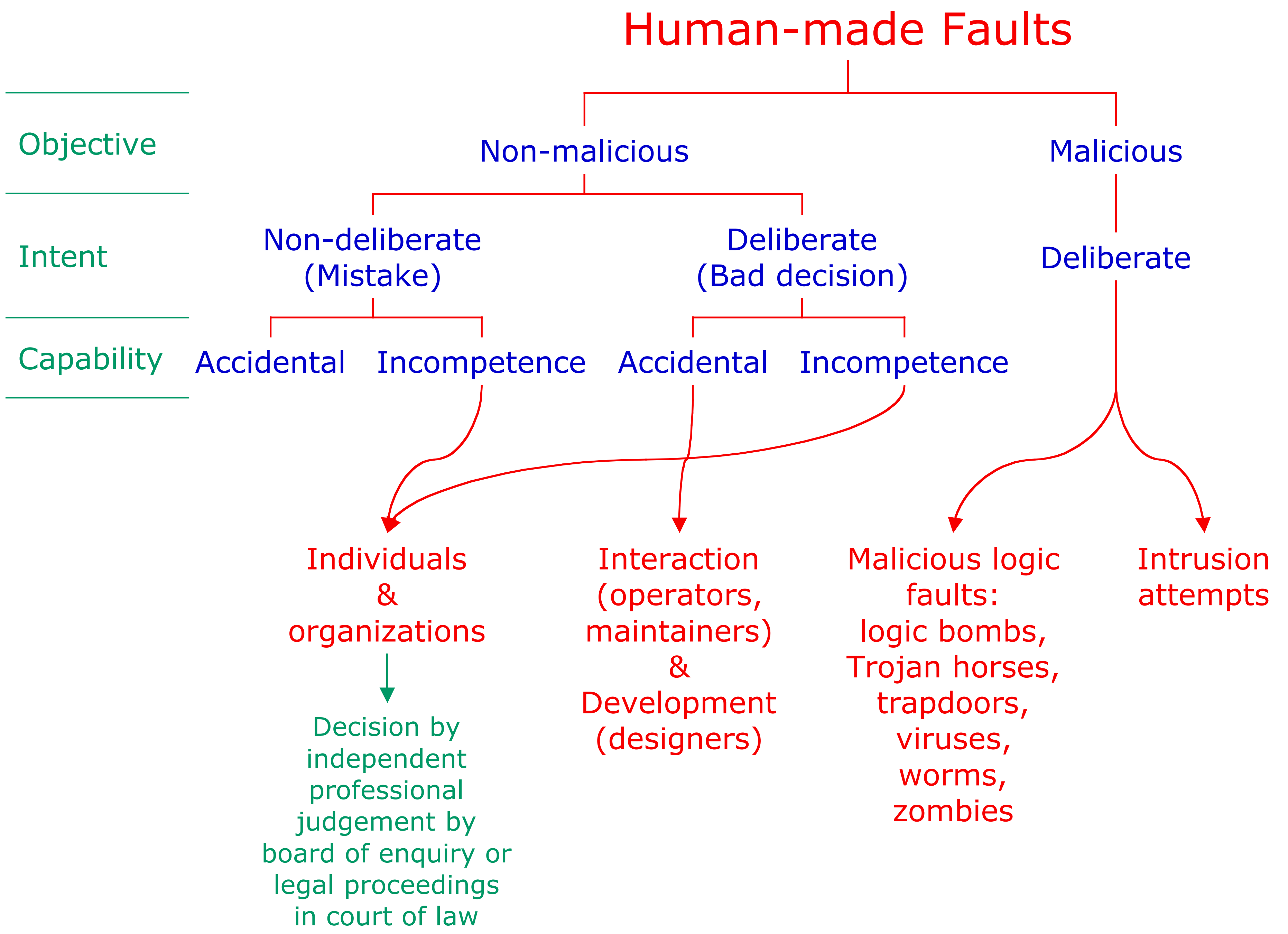
Service Threats

… Failures → Faults → Errors → Failures → Faults …

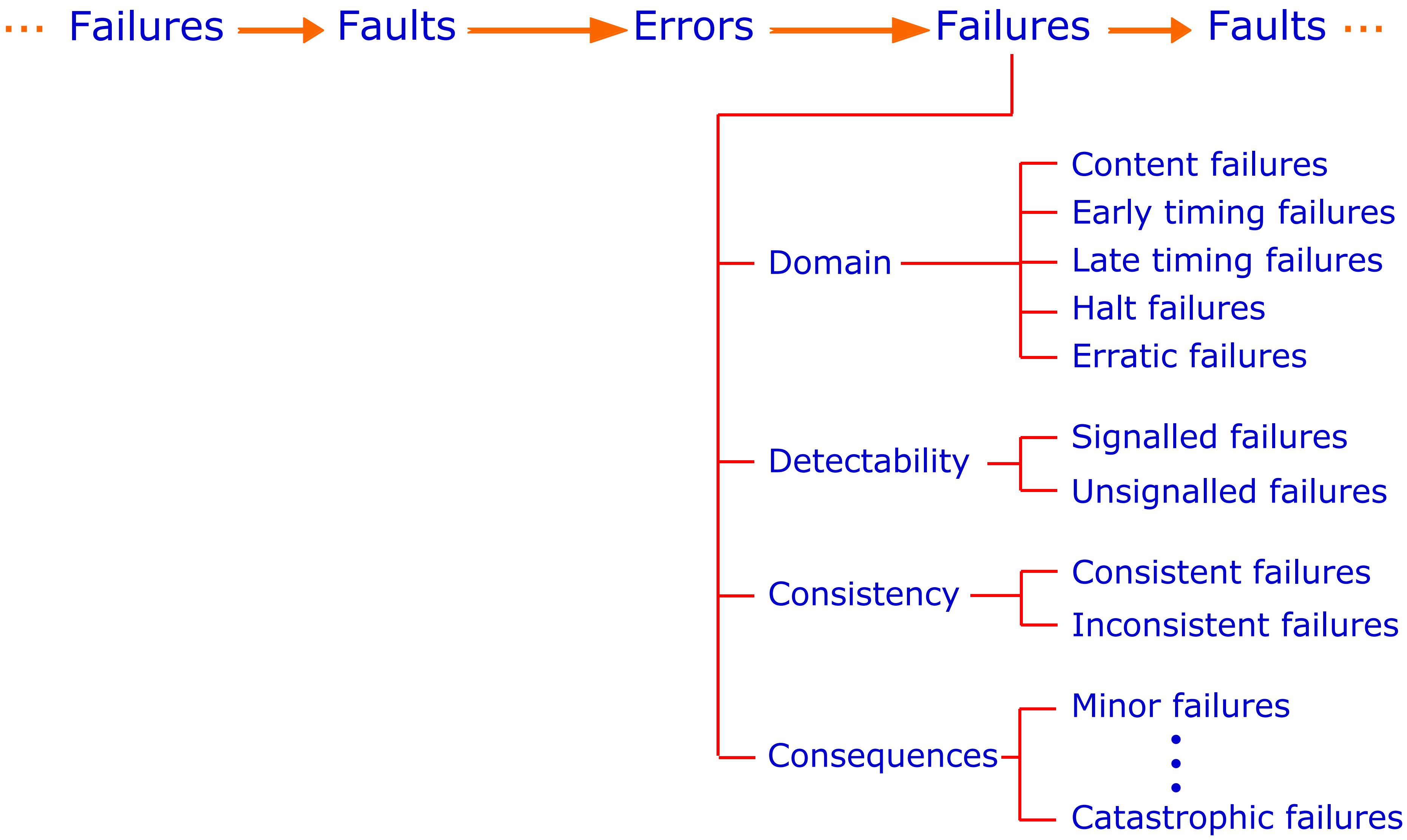
- Faults
  - Phase of creation or occurrence
    - Development faults
    - Operational faults
  - System boundaries
    - Internal faults
    - External faults
  - Phenomenological cause
    - Natural faults
    - Human-made faults
  - Dimension
    - Hardware faults
    - Software faults
  - Objective
    - Malicious faults
    - Non-malicious faults
  - Capability
    - Accidental faults
    - Incompetence faults
  - Persistence
    - Permanent faults
    - Transient faults

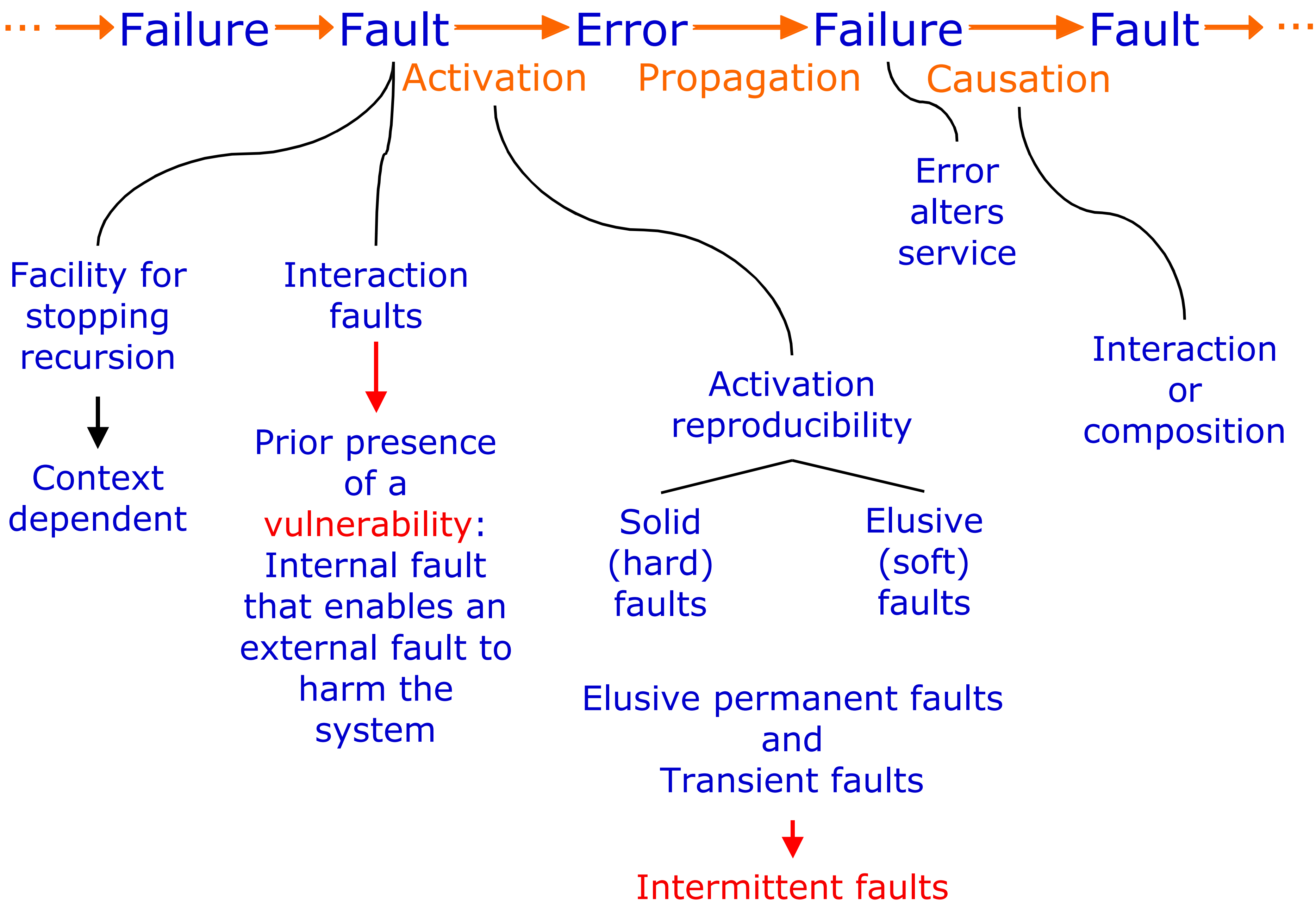


Faults
Phase of creation or occurrence
System boundaries
Phenomenological cause
Dimension
Objective
Intent
Capability
Persistence
Development
Operational
Internal
Internal
External
Human-made
Nat
Nat
Nat
Human-made
Software
Hardware
Hdw
Hdw
Hdw
Hardware
Software
Non Malicious
Mal
Mal
Non Malicious
Non Mal
Non Mal
Non Mal
Non Malicious
Mal
Mal
Non Malicious
Non Del
Del
Del
Del
Non Del
Del
Non Del
Non Del
Non Del
Non Del
Del
Del
Del
Non Del
Del
Acc Inc Acc Inc
Acc Inc Acc Inc Acc
Acc
Acc Acc Inc Acc Inc
Acc Inc Acc Inc
Per Per Per Per Per Per Per Per Per Per Per Per Tr Per Tr Tr Per Tr Tr Per Tr Per Tr Tr Per Tr Per Tr Tr Per Tr
Software Flaws
Logic Bombs
Hardw Errata
Produc Defects
Phys Deter
Physical Interference
Intrusion Attempts
V W
Input Mistakes
Development Faults
Physical Faults
Interaction Faults


Human-made Faults
Objective
Non-malicious
Malicious
Intent
Non-deliberate (Mistake)
Deliberate (Bad decision)
Deliberate
Capability
Accidental
Incompetence
Accidental
Incompetence
Individuals & organizations
Decision by independent professional judgement by board of enquiry or legal proceedings in court of law
Interaction (operators, maintainers) & Development (designers)
Malicious logic faults: logic bombs, Trojan horses, trapdoors, viruses, worms, zombies
Intrusion attempts

Service Threats





… Failure Fault Error Failure Fault …
Activation
Propagation
Causation
Facility for stopping recursion
Context dependent
Interaction faults
Prior presence of a vulnerability: Internal fault that enables an external fault to harm the system
Activation reproducibility
Solid (hard) faults
Elusive (soft) faults
Elusive permanent faults and Transient faults
Intermittent faults
Error alters service
Interaction or composition

Development failures

Development process terminates before the system is accepted for use and placed into service

Causes:

- Incomplete or faulty specifications
- Excessive number of specification changes
- Inadequate design wrt functionality or performance
- Too many development faults
- Insufficient predicted dependability
- Faulty estimates of development costs

Partial development failures

- Budget or schedule overruns
- Downgrading to less functionality, performance, dependability

Dependability and its attributes

- Definitions of dependability
  - Original definition: ability to deliver service that can justifiably be trusted
    - Aimed at generalizing availability, reliability, safety, confidentiality, integrity, maintainability, that are then attributes of dependability
    - Focus on trust, i.e. accepted dependence
    - Dependence of system A on system B is the extent to which system A's dependability is (or would be) affected by that of system B
  - Alternate definition: ability to avoid service failures that are more frequent or more severe than is acceptable
    - A system can, and usually does, fail. Is it however still dependable ? When does it become undependable ?

      ✂ ⬇

    - criterion for deciding whether or not, in spite of service failures, a system is still to be regarded as dependable
    - Dependability failure ← development fault(s)

❖Dependability vs. High Confidence vs. Survivability vs. 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)

Dependability

- Subsumes concerns in reliability, availability, safety, confidentiality, integrity, maintenability — the *attributes of dependability* — within a unified conceptual framework; enables the appropriate balance between the attributes to be addressed
- *Means for dependability* — fault prevention, fault tolerance, fault removal, fault forecasting — provide an orthogonal classification of development activities; essential for abstract and discrete systems (nonexistent or vanishing safety factor)
- Causal chain of *threats to dependability* — fault - error - failure
  - Central to understanding and mastering various threats likely to affect a system
  - Provides for a unified presentation of those threats, though preserving their specificities via the various classes
- Rigorous terminology — not just definitions: a model

  abstraction   structuration   recursion

  - Avoiding intellectual confusion(s)
  - Focusing on scientific problems and technical choices

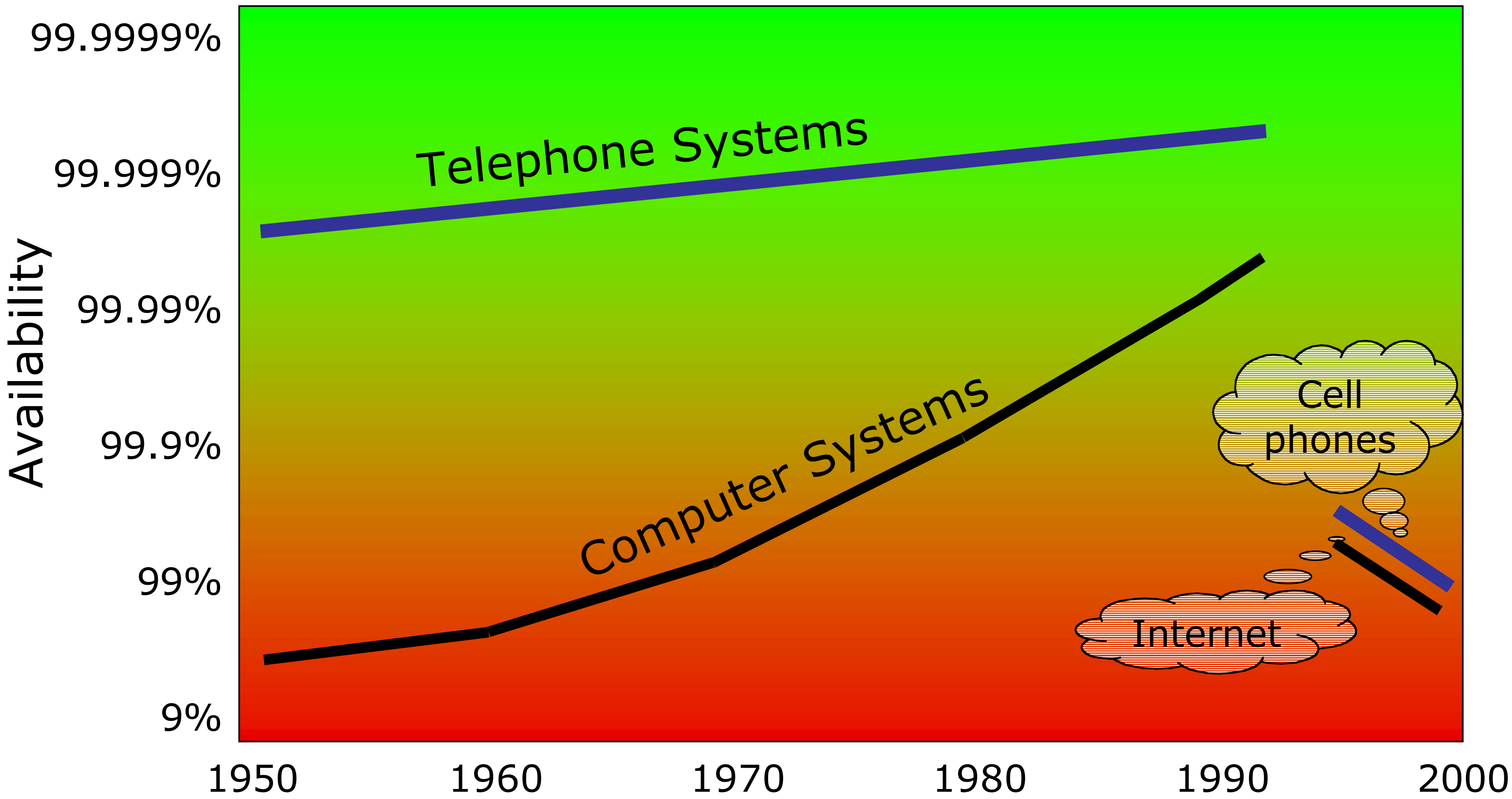
❖Service failures

	Faults			Failures		Availability/ Reliability	Safety	Confidentiality
	Physical	Development	Interaction	Localized	Distributed			
June 1980: False alerts at the North American Air Defense (NORAD)	✔			✔		✔		
April 1981: First launch of the Space Shuttle postponed		✔		✔		✔		
June 1985 - January 1987: Excessive radiotherapy doses (Therac-25)		✔		✔			✔	
August 1986 - 1987: the "wily hacker" penetrates several tens of sensitive computing facilities		✔	✔	✔				✔
November 1988: Internet worm		✔	✔		✔	✔		
15 January 1990: 9 hours outage of the long-distance phone in the USA		✔			✔	✔		
February 1991: Scud missed by a Patriot (Dhahran, Gulf War)		✔	✔	✔		✔	✔	
November 1992: Crash of the communication system of the London ambulance service		✔	✔		✔	✔	✔	
26 and 27 June 1993: Authorization denial of credit card operations in France	✔	✔			✔	✔		
4 June 1996: Failure of Ariane 5 maiden flight		✔		✔		✔		
13 April 1998: Crash of the AT&T data network		✔	✔		✔	✔		
February 2000: Distributed denials of service on large Web sites		✔	✔		✔	✔		
May 2000: Virus I love you		✔	✔		✔	✔		
July 2001: Worm Code Red		✔	✔		✔	✔		
July 2001: Worm Sircam		✔	✔		✔			✔
August 2003: Propagation of the electricity blackout in the USA and Canada		✔	✔		✔	✔		

Non-malicious faults

Number of failures by causes [consequences and outage durations highly application-dependent]	Dedicated computer systems (e.g., transactions, electronic switching, Internet back-end servers)		Larger, controlled systems (e.g., commercial airplanes; telephone network; Internet front-end servers for web applications)	
Faults	Rank	Proportion	Rank	Proportion
Physical internal	3	~ 10%	2	15-20%
Physical interaction	3	~ 10%	2	15-20%
Human-made interaction *	2	~ 20%	1	40-50%
Development	1	~ 60%	2	15-20%

* Root analysis evidences that human-made interaction faults often can be traced to development faults



From J. Gray, *Dependability in the Internet era*

- Complexity
- Economic pressure

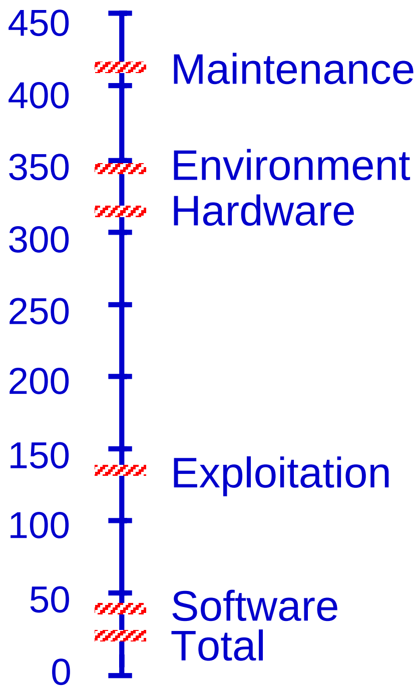
« faster, cheaper, badder »

Availability	Outage duration/yr
0,999999	32s
0,99999	5mn 15s
0,9999	52mn 34s
0,999	8h 46mn
0,99	3j 16h
0,9	36j 12h

Tandem fault tolerant systems

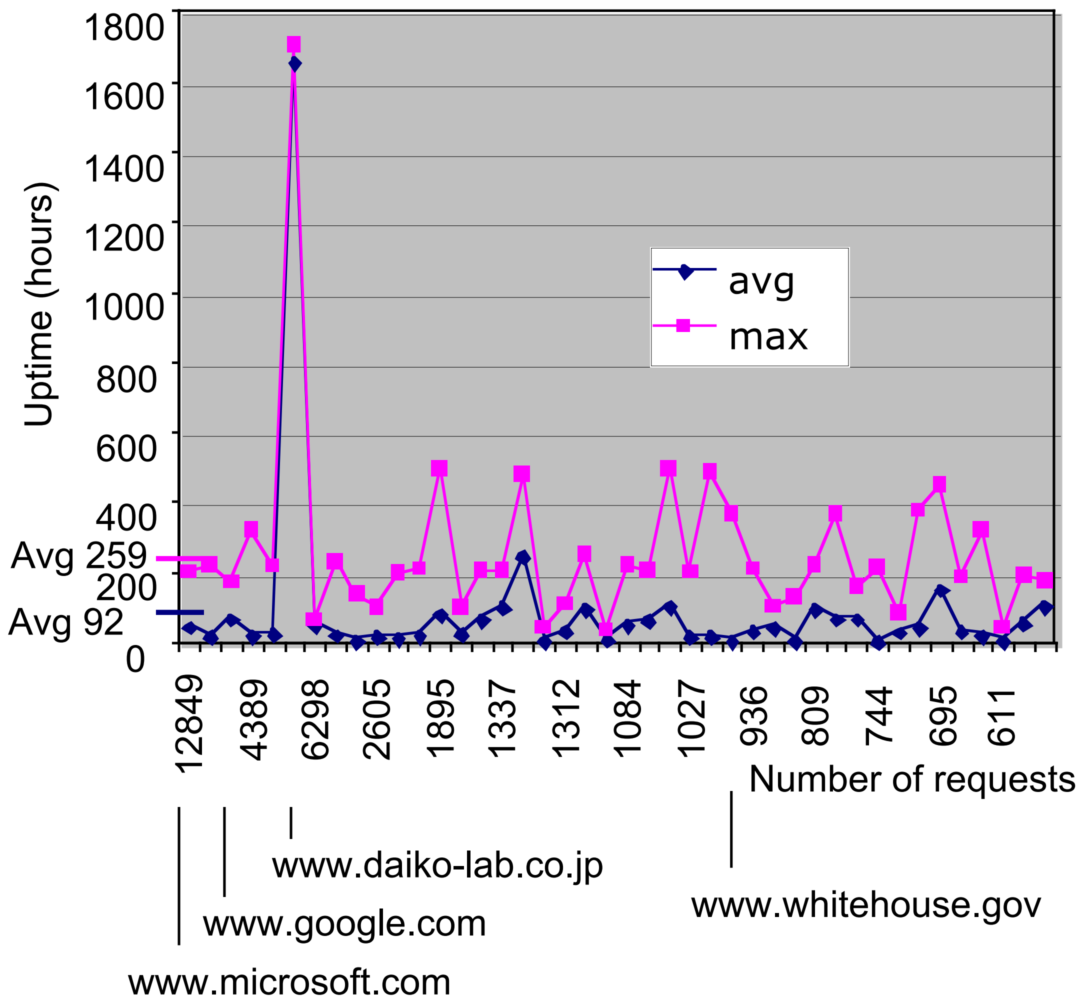
	Number	Duration (yrs)
Clients	2000	7000
Systems	9000	30000
Processors	25500	80000
Disks	74000	200000
Reported outages		438
MTBF System		21 yrs

MTBF (yrs)

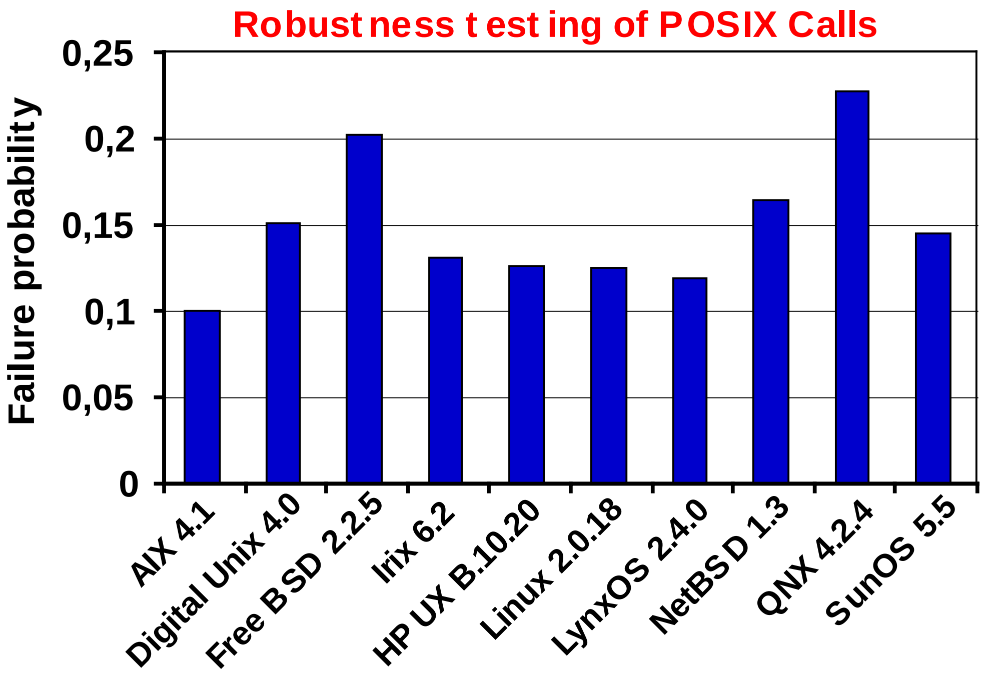


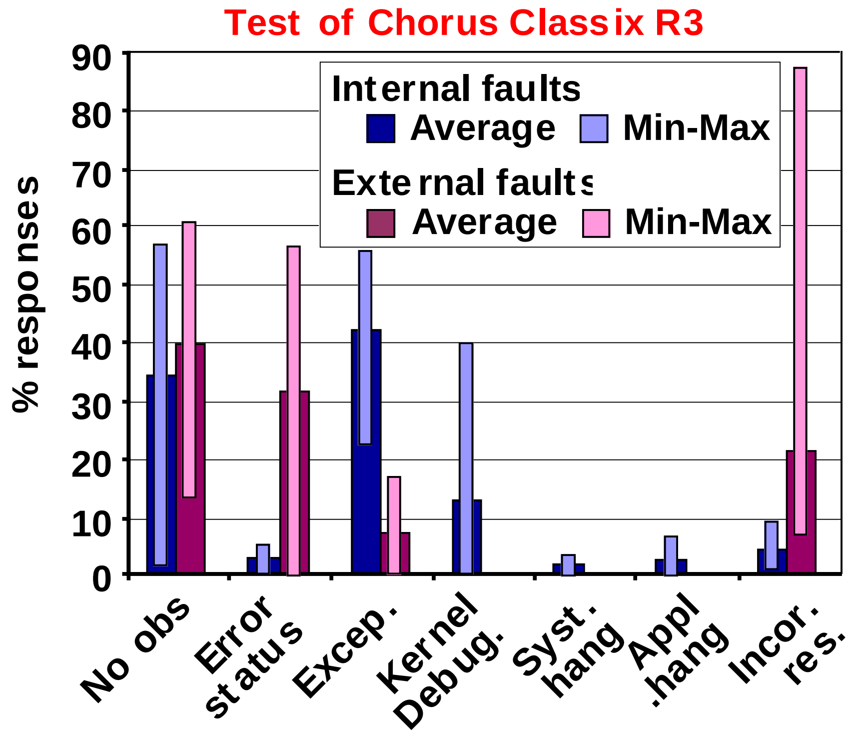
NetCraft — Uptime statistics

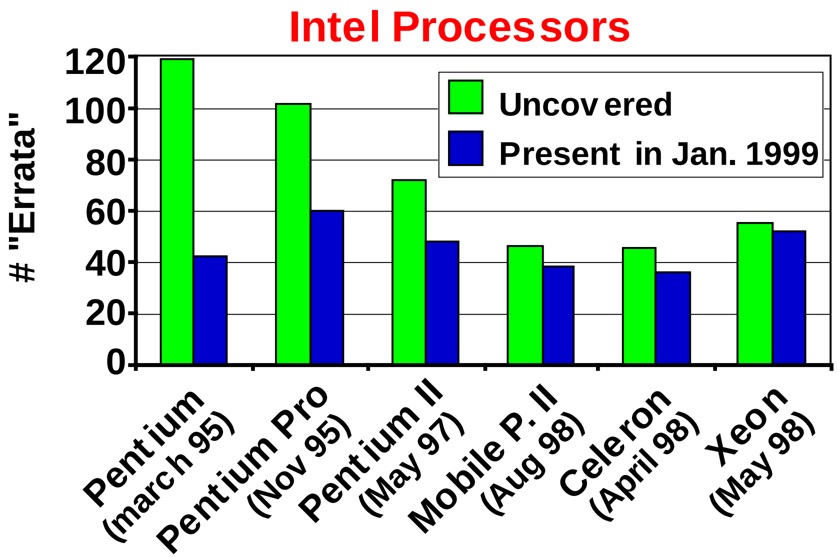
Top 50 most requested sites

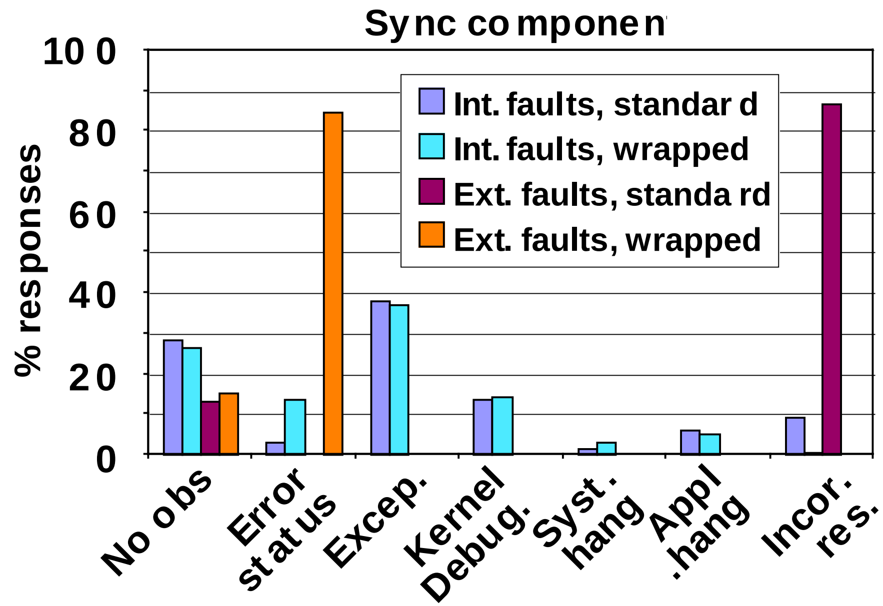


(C)OTS data: OS and hardware



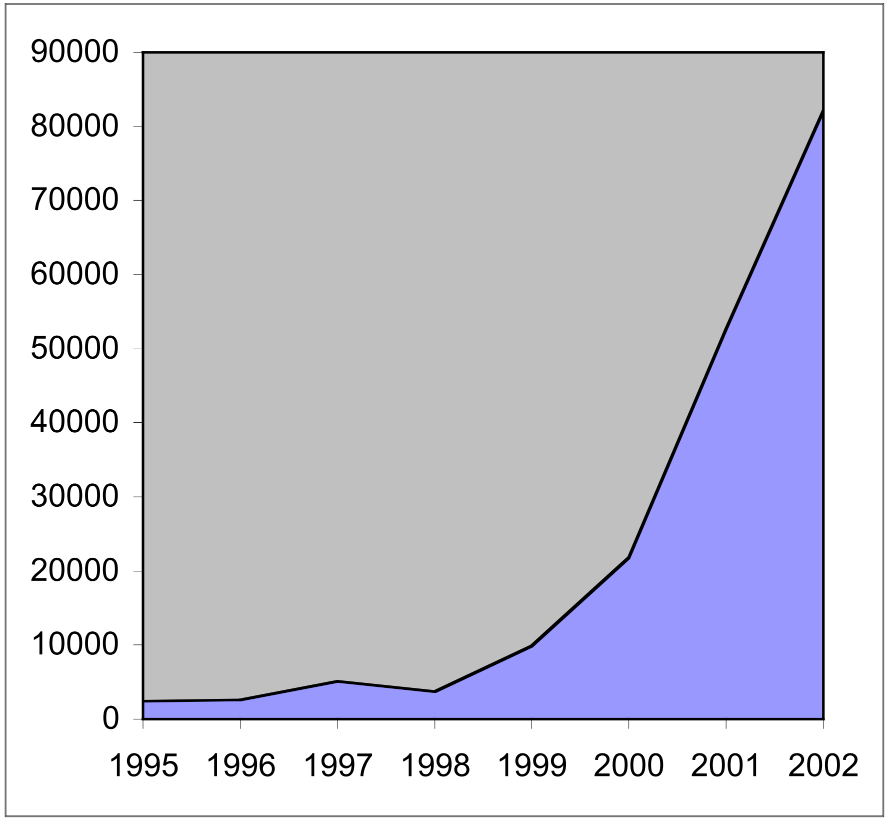




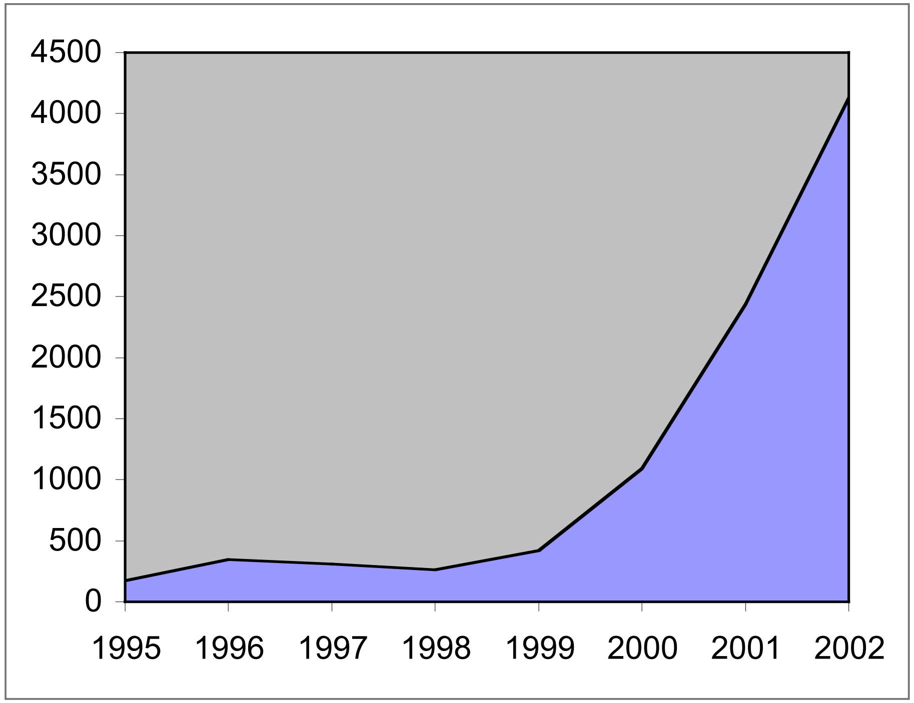


Malicious faults: statistics from SEI/CERT

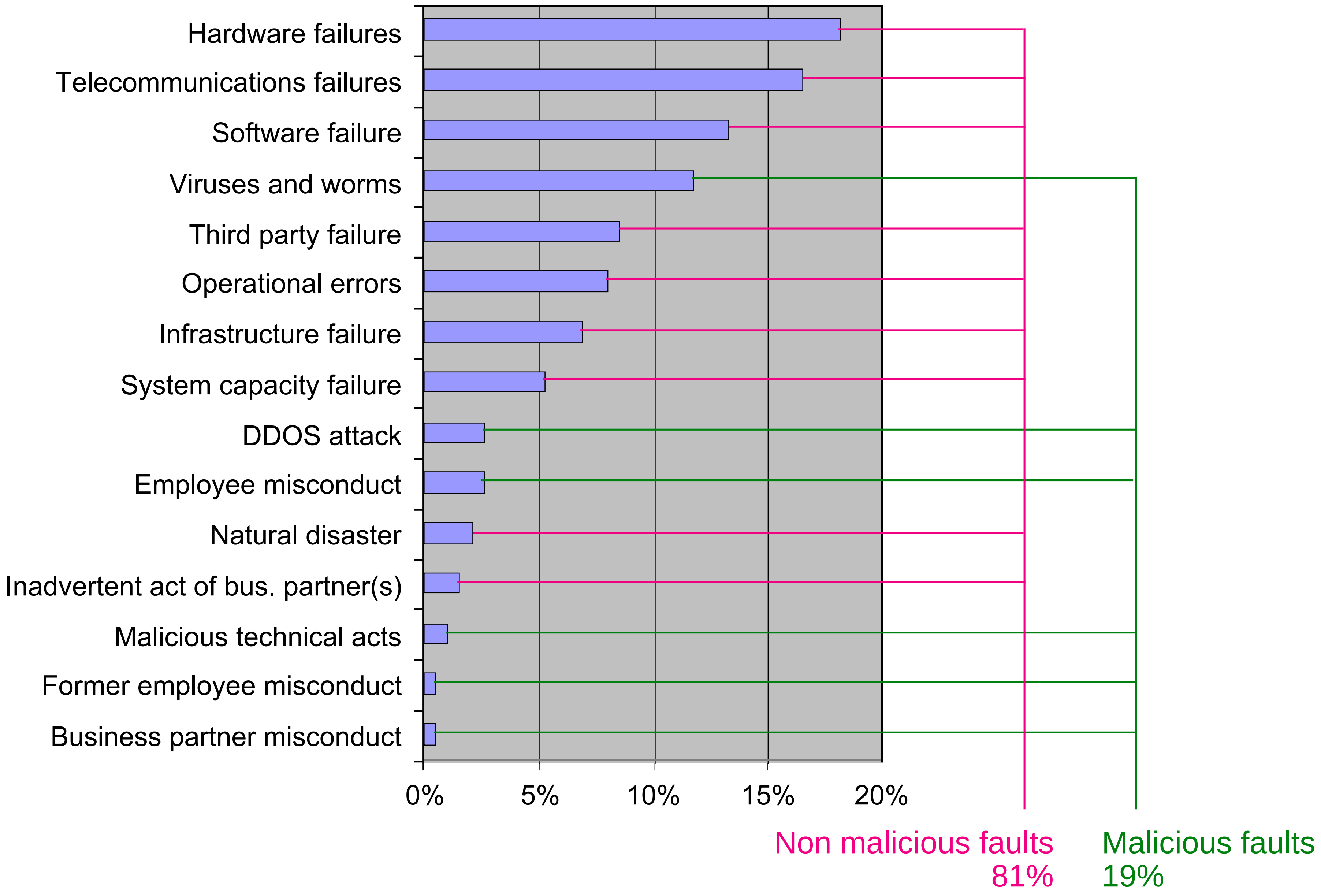
Reported incidents



Reported vulnerabilities

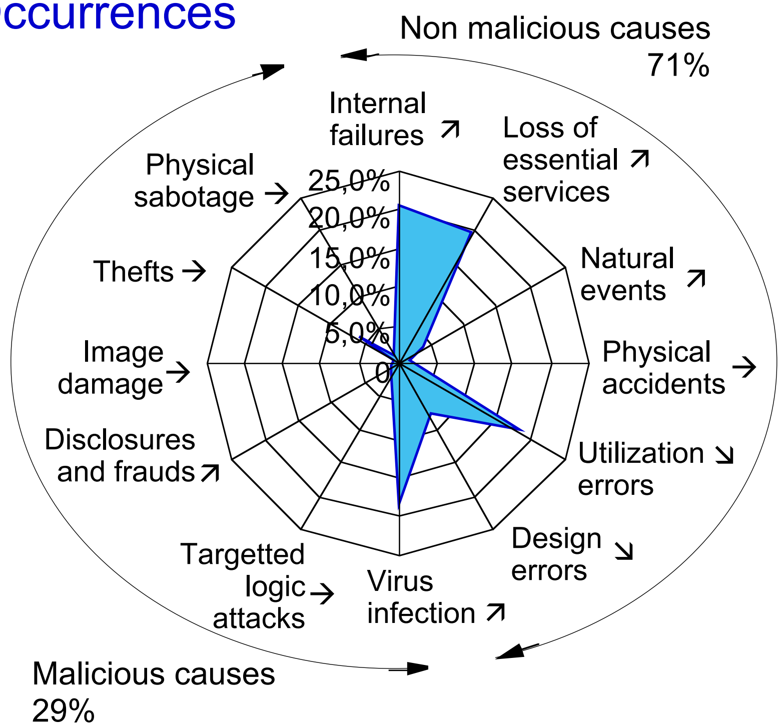


Global Information Security Survey 2003 — Ernst & Young

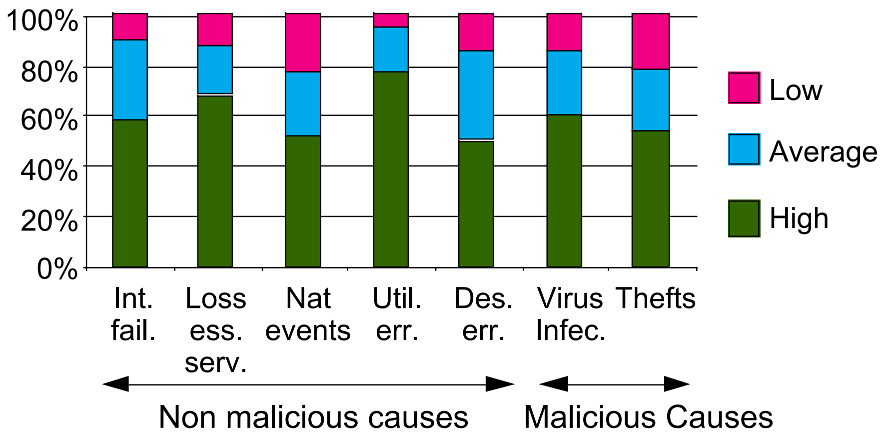


Yearly survey on computer damages in France — CLUSIF (2000, 2001, 2002)

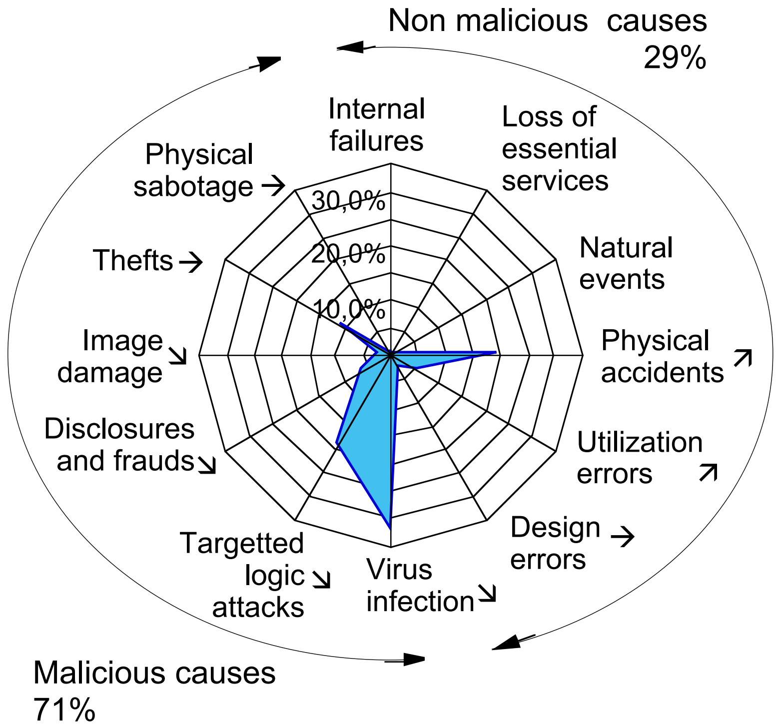
Occurrences



Occurrence impact



Risk perception



3 year trends
→ stable
↗ increase
↘ decrease

❖Development failures

	1994	2002
Number of surveyed projects	8,380	13,522
Successful projects (completed on-time and on-budget, with all features and functions as initially specified)	16%	34%
Challenged projects (completed and operational but over-budget, over the time estimate, and offers fewer features and functions than originally specified)	53%	51%
Canceled projects	31%	15%
Overruns for challenged projects	89%	82%
Left functions for challenged projects	61%	52%
Total estimated budget for software projects in the USA, in G$	250	225
Estimated lost value for software projects in the USA, in G$	81	38

From Standish Group (*Chaos reports*)

High dependability for safety-critical or dedicated systems

Avionics, railway signalling, nuclear control, etc.

Transaction processing, back-end servers, etc.

Scalability of dependability?

Continuous complexity growth (web-based applications, networked embedded systems)

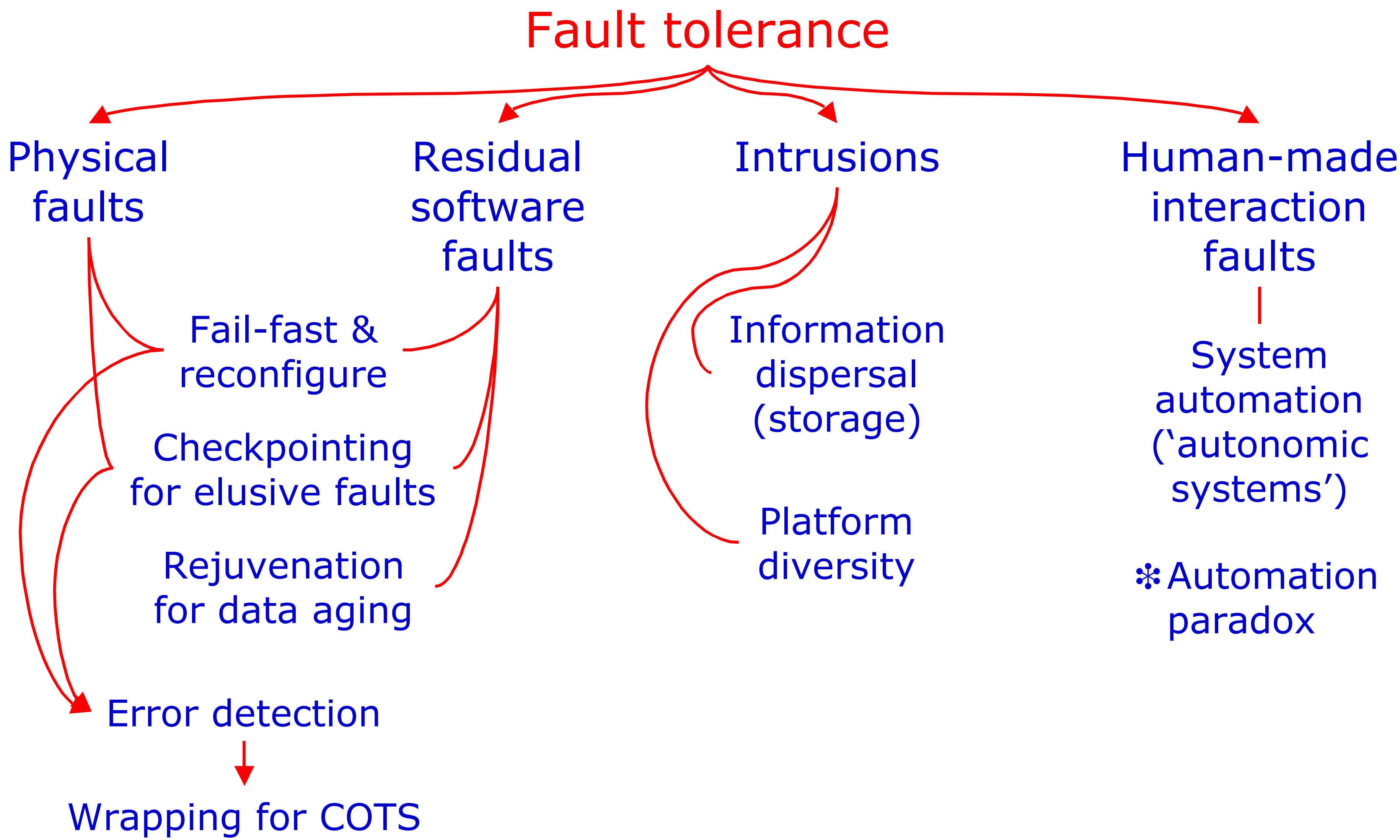
↓

In addition to fault removal,

Generalization of

fault tolerance

- Physical faults
- Residual software faults
- Intrusions
  ↓
  Vulnerabilities [some unavoidable for usability]
- Human-made interaction faults: Administration, configuration, maintenance faults



Fault tolerance assessment

Coverage demonstration, by analysis (incl. formal) and by experiments (representative fault injection)