VDE VERLAG

Zuverlässigkeit und Entwurf

231

ITG-Fachbericht

5. GI/GMM/ITG-Fachtagung
vom 27. bis 29. September 2011
in Hamburg-Harburg

mit CD-ROM

Gesellschaft für Informatik (GI)
VDE/VDI-Gesellschaft Mikroelektronik, Mikrosystem- und Feinwerktechnik (GMM)
Informationstechnische Gesellschaft im VDE (ITG)

5. GI/GMM/ITG Fachtagung "Zuverlässigkeit und Entwurf" [Reliability and Design]

September 27-29, 2011 — Hotel Panorama, Hamburg-Harburg, Germany

Dependable Computing and Assessment of Dependability

Jean Arlat

[http://homepages.laas.fr/arlat]

LAAS-CNRS

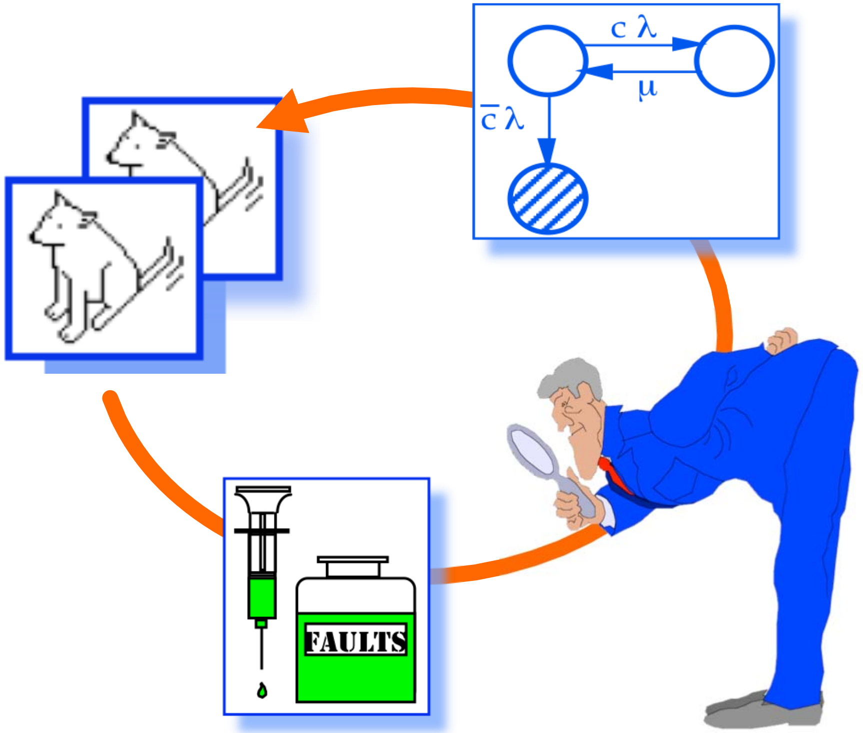
INSTITUT CARNOT LAAS CNRS

cnrs dépasser les frontières

Université de Toulouse

Outline

- Dependable Computing
  - Basic Definitions and Terminology
  - Fault Tolerance
- Dependability Assessment
  - Experimental Validation of Fault-Tolerant Computing Systems
  - Dependability Benchmarking of Computers Systems and Components



About Dependability

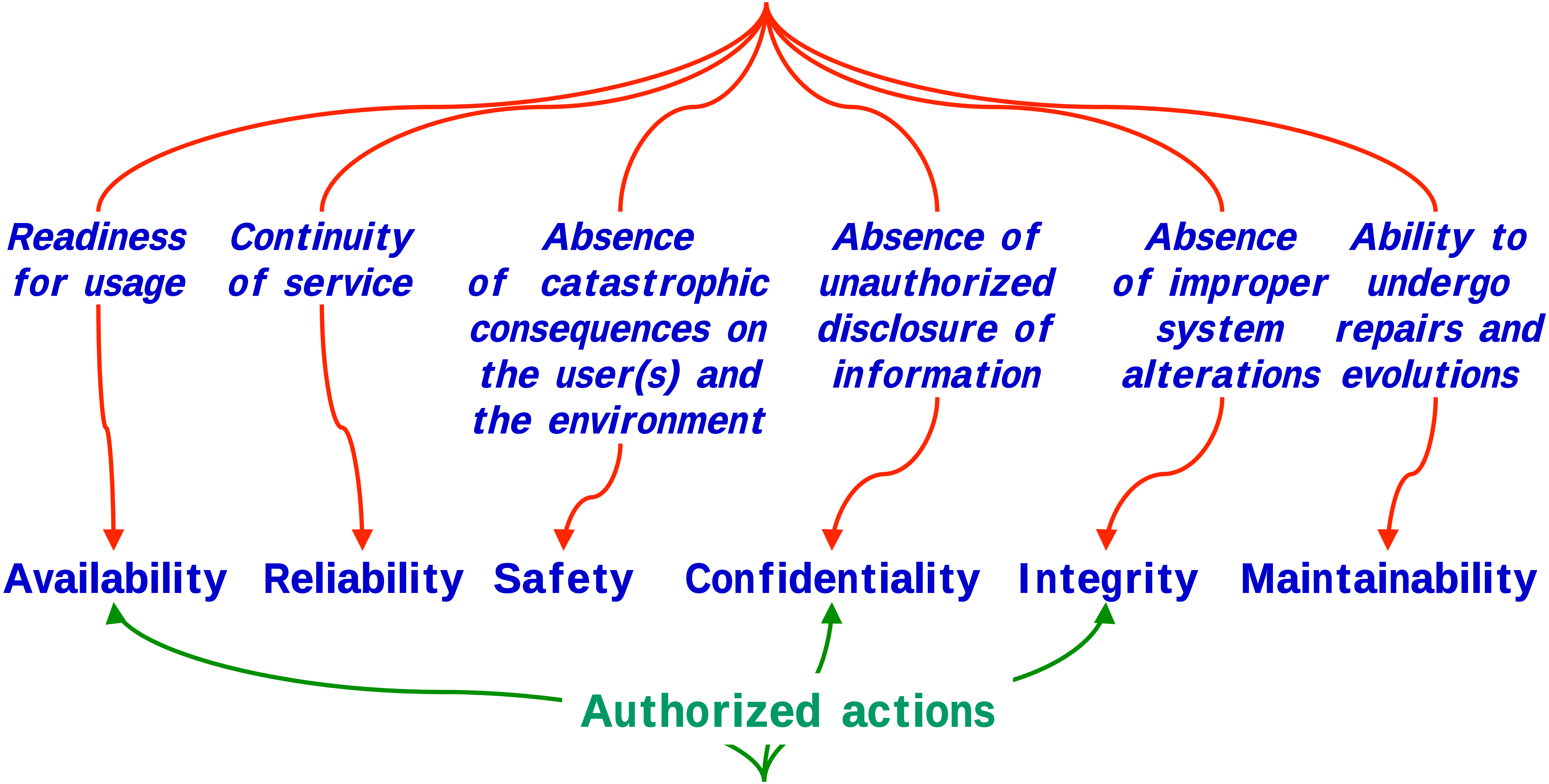
Dependability: ability to deliver service that can justifiably be trusted

- 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
- System 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
- Failure modes: the ways in which a system can fail, ranked according to failure severities

Dependability: ability to avoid failures that are more frequent or more severe than is acceptable to the user(s)

- When 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

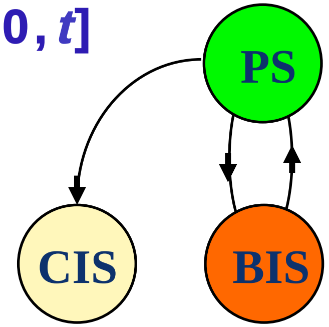
Security

Absence of unauthorized access to, or handling of, system state

Dependability Measures

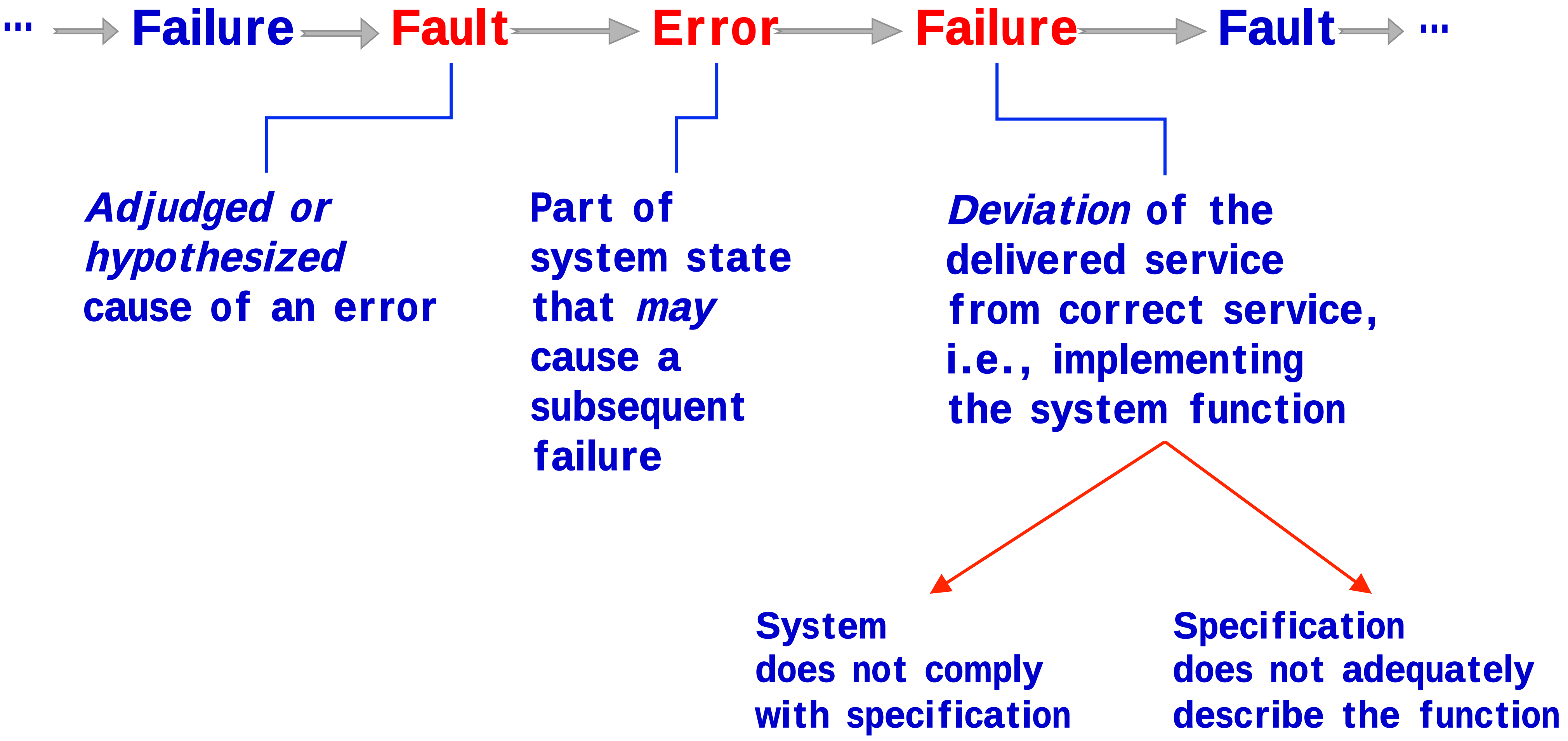
- ***Availability*** - quantifies the alternation between deliveries of proper and improper service
  - $A(t)$ = 1 if service is proper at time t, 0 otherwise
- ***Reliability*** - continuous delivery of proper service
  - $R(t)$: probability that a system delivers proper service throughout $[0,t]$
- ***Safety*** - time to catastrophic failure
  - $S(t)$: probability that no catastrophic failures occur during $[0,t]$

    [Analogous to reliability, but concerned with catastrophic failures]



- ***Time to Failure*** - time to failure from last restoration [Expected value of this measure is referred to as *MUT - Mean Up Time*]
- ***Maintainability*** - time to restoration from last experienced failure. [Expected value is referred to as *MDT - Mean Down Time*]
- ***Coverage*** - probability that, given a fault, the system can tolerate the fault and continue to deliver proper service

The "fault-error-failure" sequence



… → Failure → Fault → Error → Failure → Fault → …

Adjudged or hypothesized cause of an error

Part of system state that *may* cause a subsequent failure

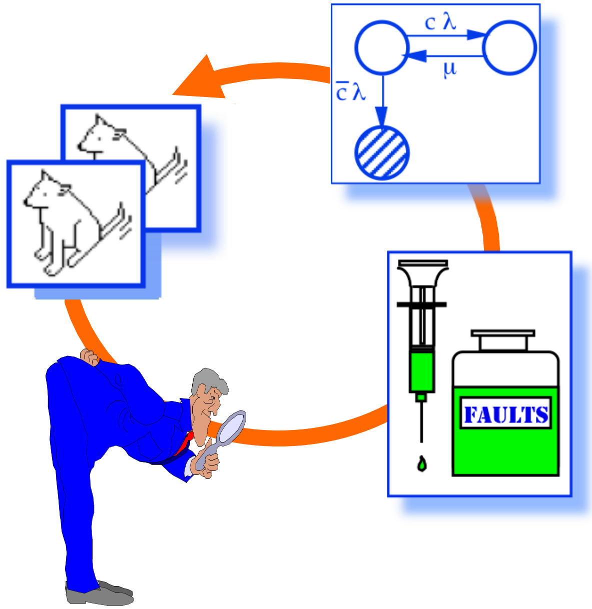
Deviation of the delivered service from correct service, i.e., implementing the system function

System does not comply with specification

Specification does not adequately describe the function

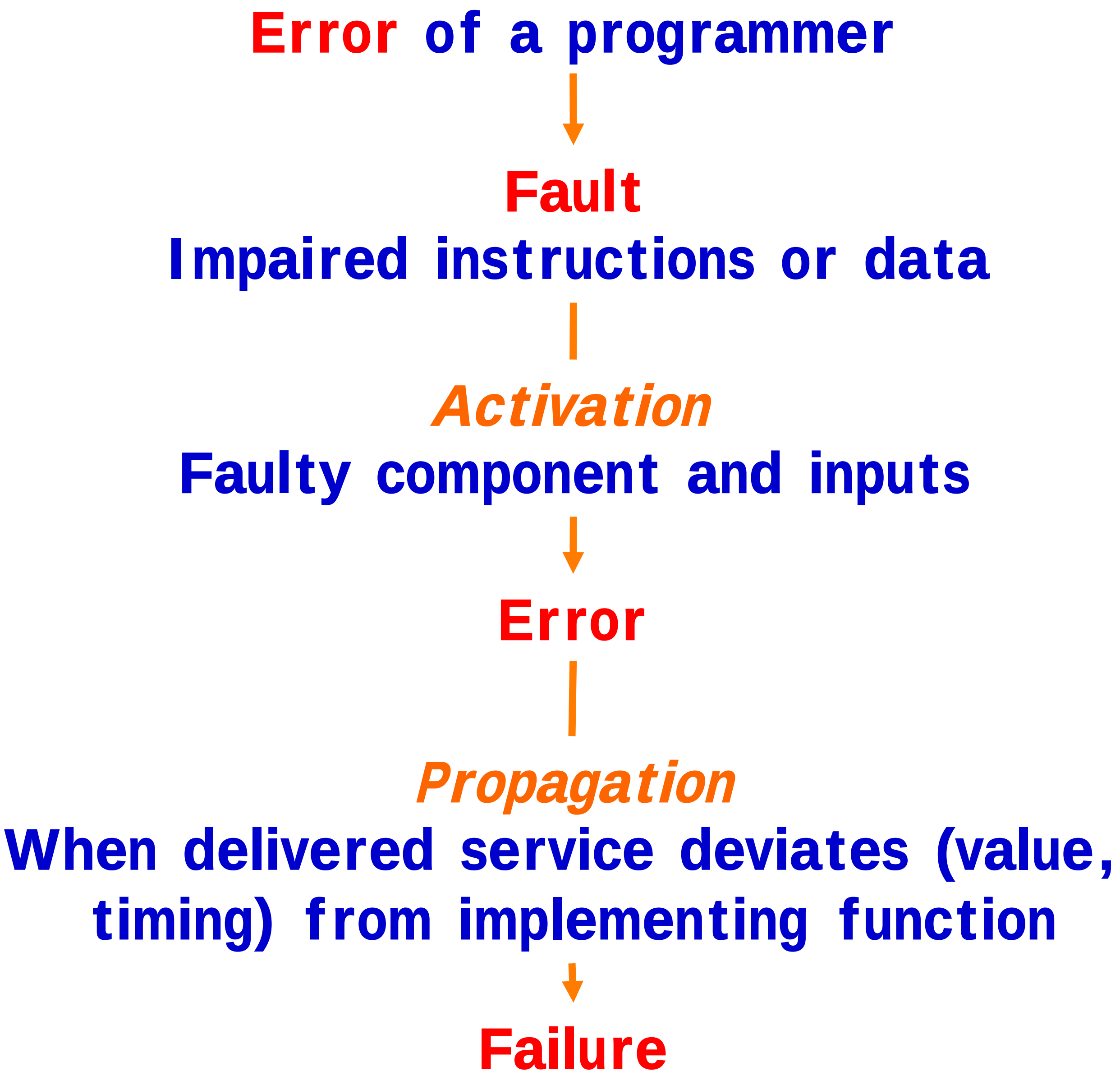
The "Dependability Tree" *

- **Dependability**
  - *Attributes*
    - Availability
    - Reliability
    - Safety
    - Confidentiality
    - Integrity
    - Maintainability
    - Security
  - *Means*
    - Procurement
      - Fault Prevention
      - Fault Tolerance
    - Assessment
      - Fault Removal
      - Fault Forecasting
  - *Threats*
    - Faults
    - Errors
    - Failures



* A. Avižienis, J.-C. Laprie, B. Randell, C. Landwehr
Basic Concepts and Taxonomy of Dependable and Secure Computing
IEEE TDSC, 1 (1), pp. 11-33, Jan.-March 2004

Software Fault Pathology



Error of a programmer

Fault

Impaired instructions or data

Activation

Faulty component and inputs

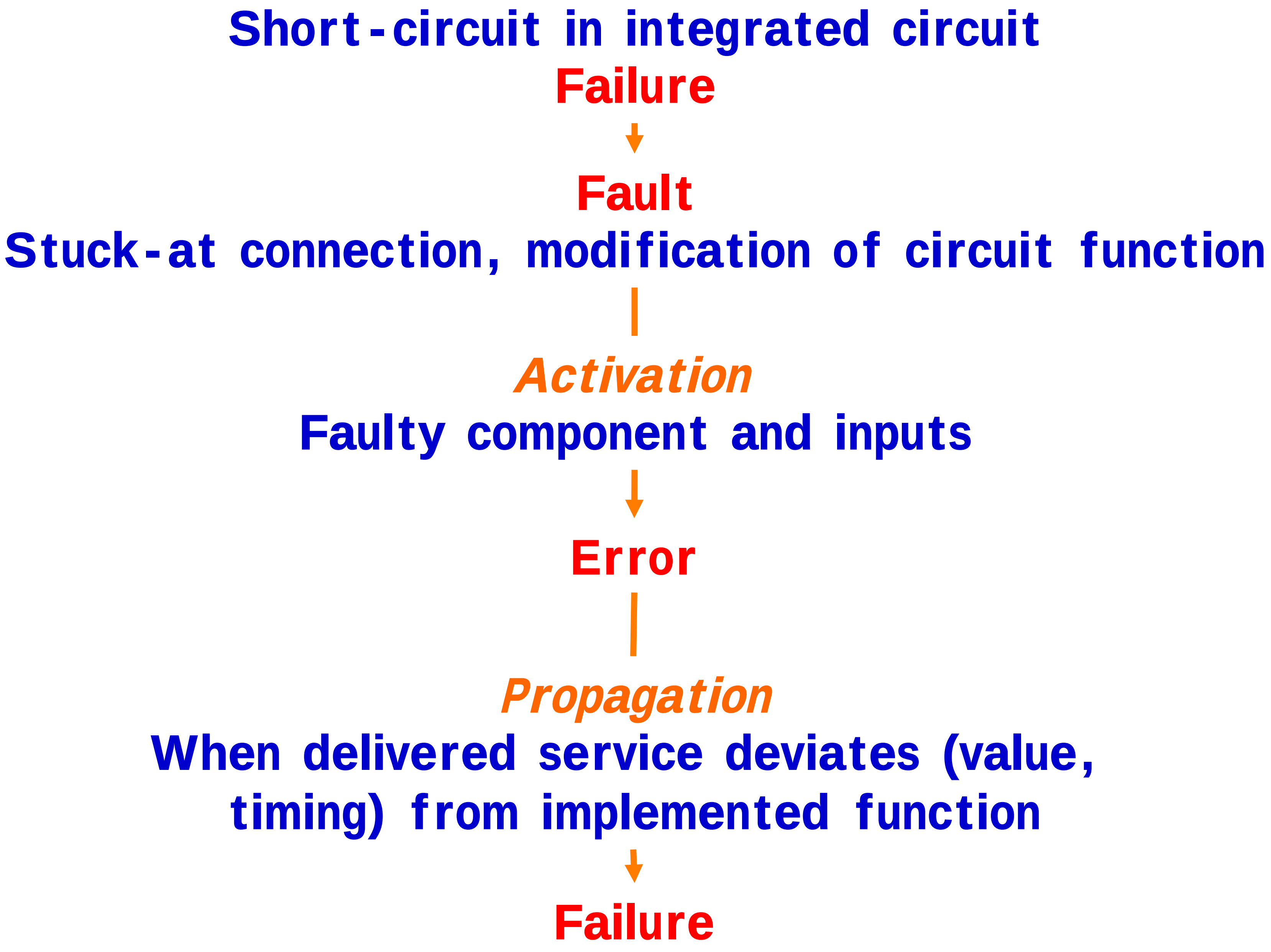
Error

Propagation

When delivered service deviates (value, timing) from implementing function

Failure

Hardware Fault Pathology



Short-circuit in integrated circuit

Failure

Fault

Stuck-at connection, modification of circuit function

Activation

Faulty component and inputs

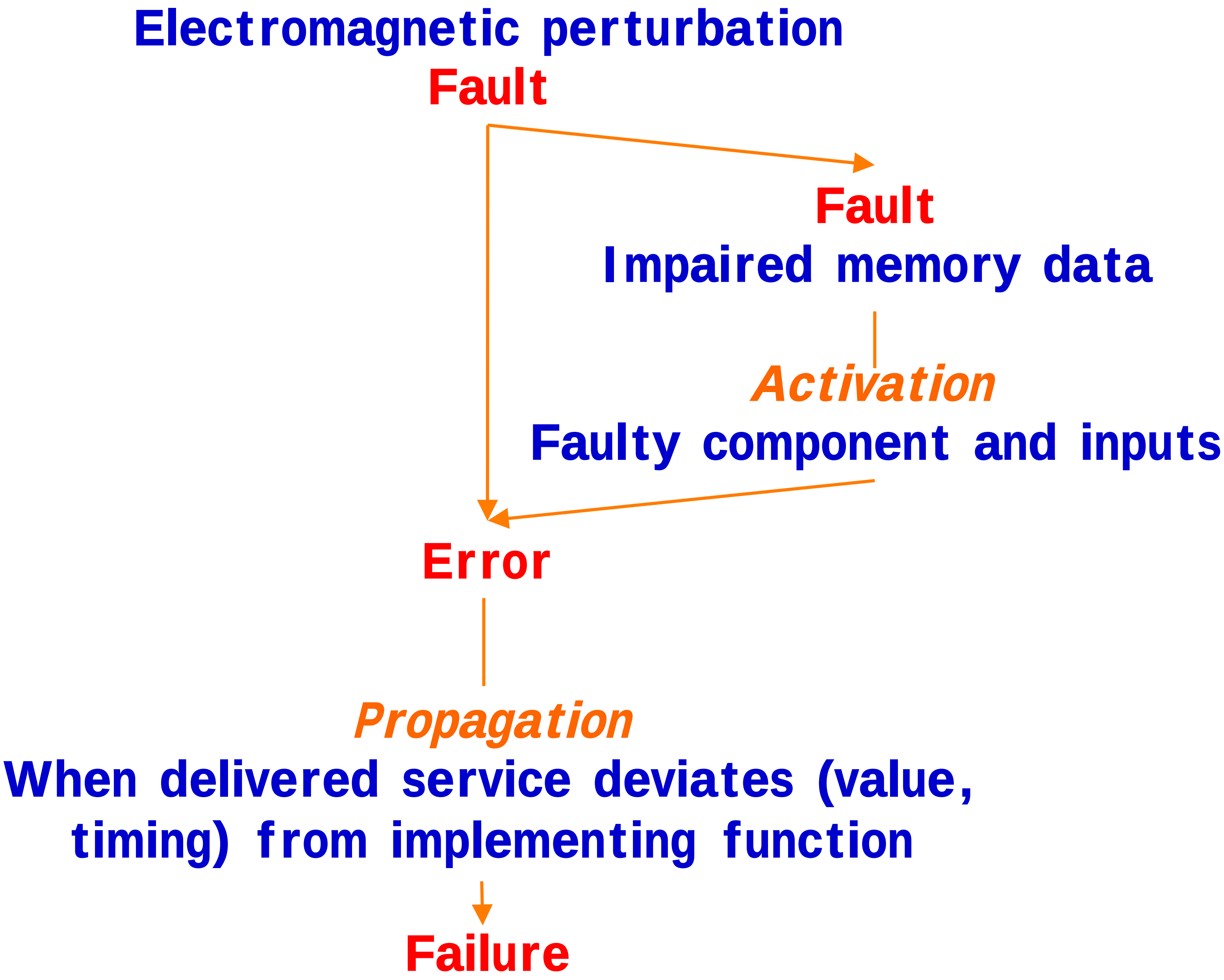
Error

Propagation

When delivered service deviates (value, timing) from implemented function

Failure

Environment Fault Vulnerability



Electromagnetic perturbation

Fault

Fault

Impaired memory data

Activation

Faulty component and inputs

Error

Propagation

When delivered service deviates (value, timing) from implementing function

Failure

Fault Tolerance

Deliver service implementing system function in spite of faults

- **Error detection**: identification of error presence
- **System recovery**: transformation of erroneous state in a state free from detected error and from fault that canbe activated again
    - **Error handling:** error removal from system state, if possible before failure occurrence
    - **Fault handling** : avoiding fault(s) to be activated again

Error detection

- **Concurrent detection**, during service delivery
  Addition of error detection mechanisms in component
  → **Self-checking component**
- **Preemptive detection**: service delivery suspended, search for latent errors and dormant faults

Error handling

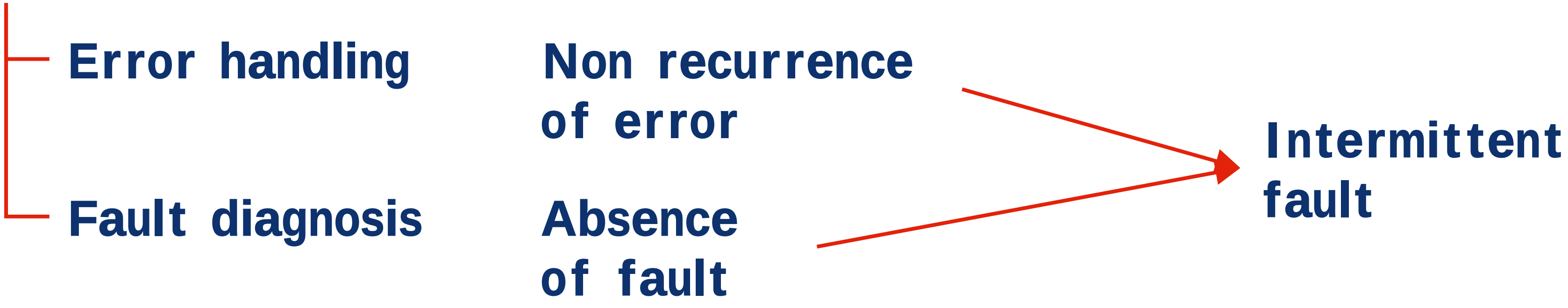
- **Backward Recovery (*Rollback*)**: brings the system back into a state saved prior to error occurrence
  Saved state = recovery point
- **Forward Recovery (*Rollforward*)**:
  search for a new state (free from detected error) and resume operation (possibly in degraded mode)
- **Compensation**: erroneous state contains enough redundancy for enabling error masking

Fault Handling

- **Diagnosis**: identifies and records the error cause(s), according to localisation and category
- **Isolation**: performs physical or logical exclusion of the fauty component(s) from further contribution to service delivery, i.e., makes the fault(s) 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

☞ Intermittent faults

- Isolation and reconfiguration not necessary
- Identification

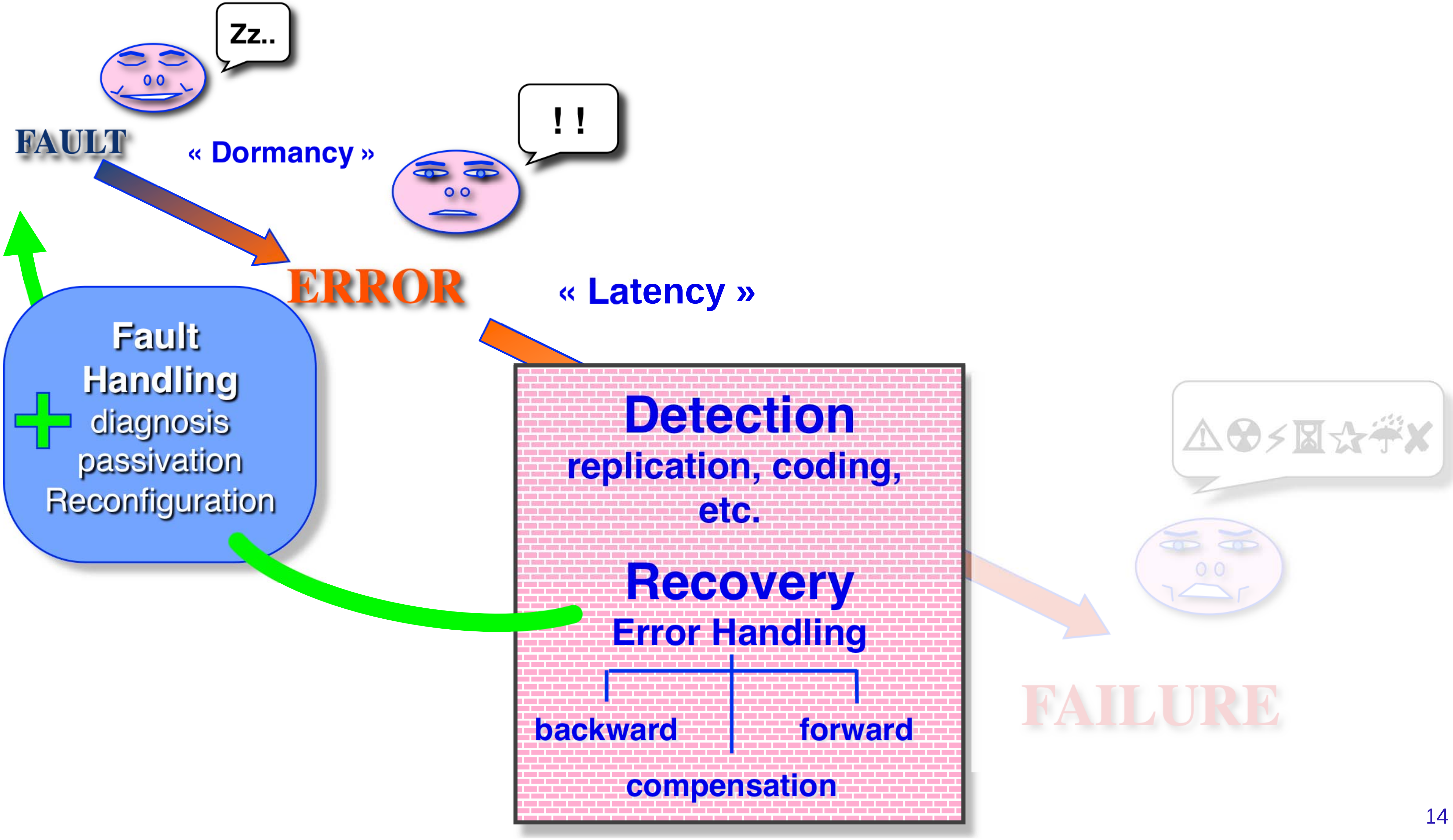


Error handling — Non recurrence of error

Fault diagnosis — Absence of fault

→ Intermittent fault

Fault Tolerance



FAULT

« Dormancy »

ERROR

« Latency »

Fault Handling

diagnosis
passivation
Reconfiguration

Detection

replication, coding, etc.

Recovery

Error Handling

backward

forward

compensation

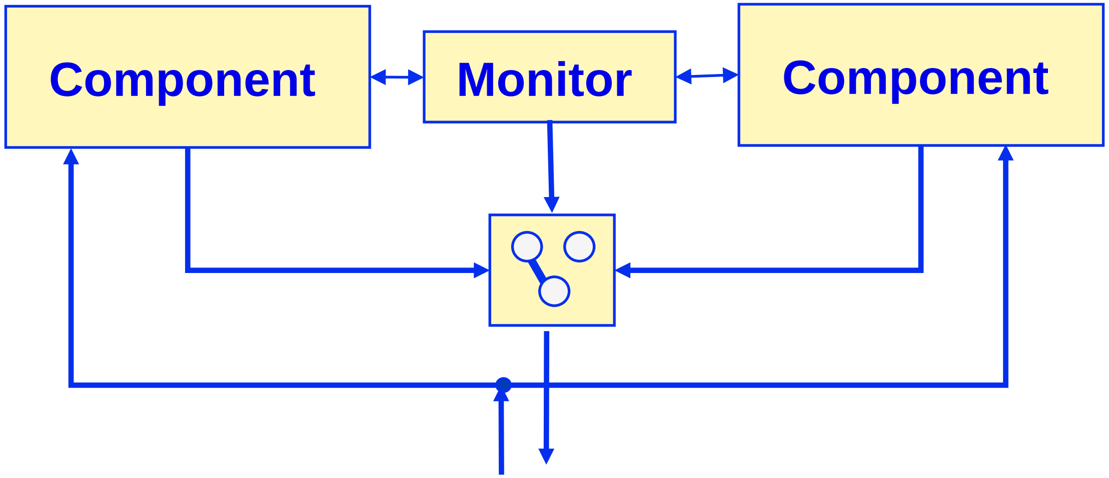
FAILURE

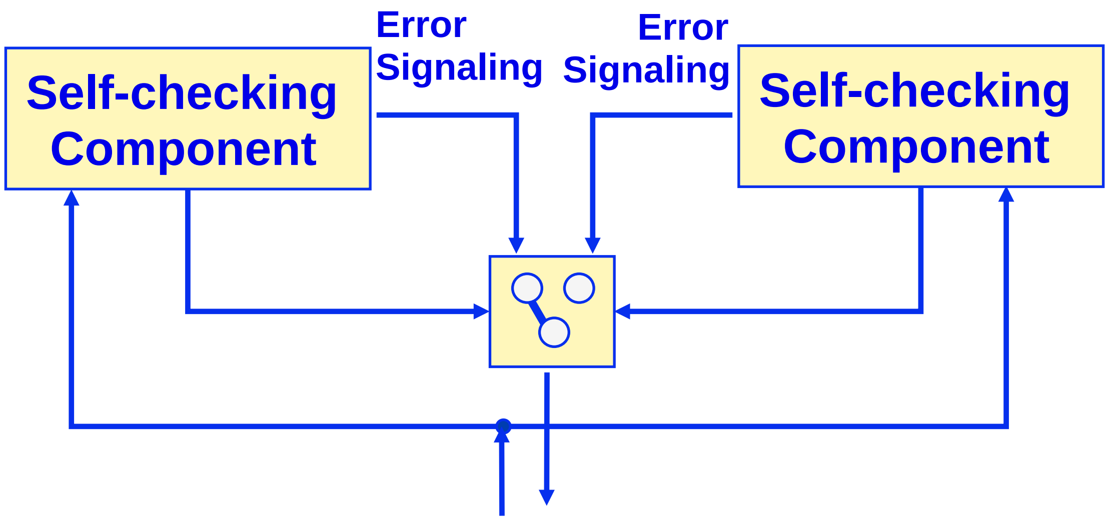
Impact of Fault Tolerance

Dependability ≈ 1 - Pr{fault} × Pr{error/fault} × Pr{failure/error}

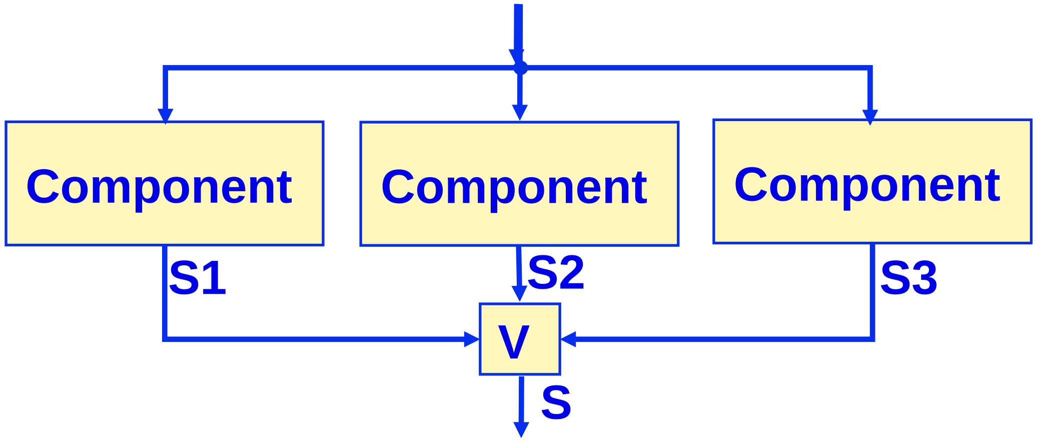
↓ System Impairments →	Fault	Error/Fault	Failure/Error
Non Fault-Tolerant (NFT)	Pr$_{NFT}${fault}	Pr$_{NFT}${error/fault}	Pr$_{NFT}${failure/error}
	∧	∧|	∨ ∨
Fault-Tolerant (FT)	Pr$_{NT}${fault}	Pr$_{FT}${error/fault}	Pr$_{FT}${failure/error}

Dynamic Redondancy (Active Duplex)





Static Redundancy: Triple Modular Redundancy

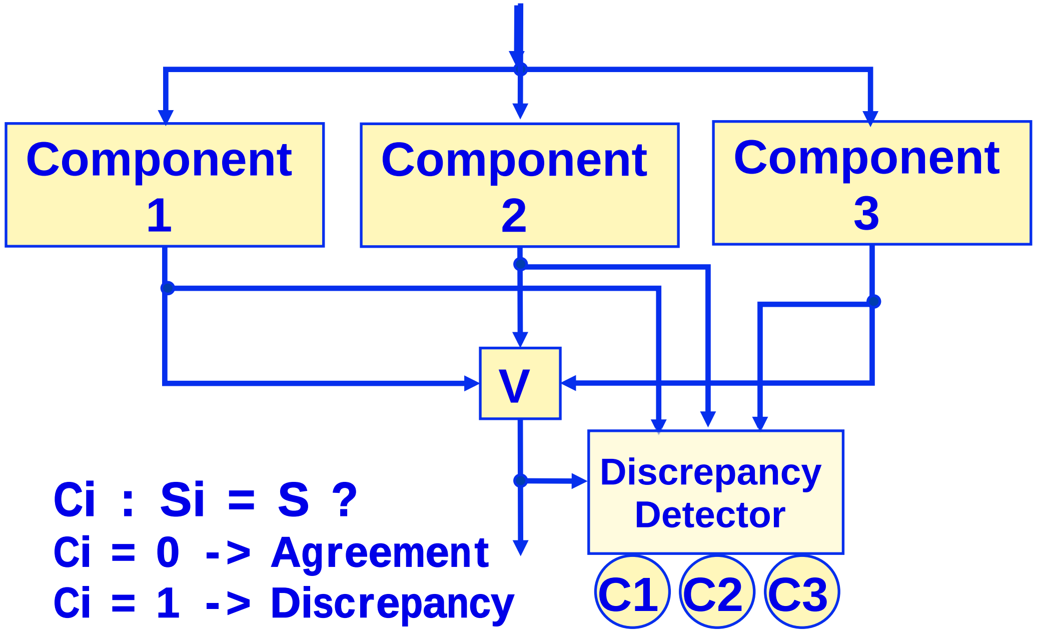


S = MAJ (S1,S2,S3)

- If S1=S2=S3=X, -> S=X
- If S1=X, S2=S3=Y
  Or S2=X; S1=S3=Y
  Or S3=X, S1=S2=Y, -> S=Y
- Either, Failure

S1, S2, S3 = Boolean variable

$S=(S1\cap S2)\cup(S2\cap S3\cup(S1\cap S3)$

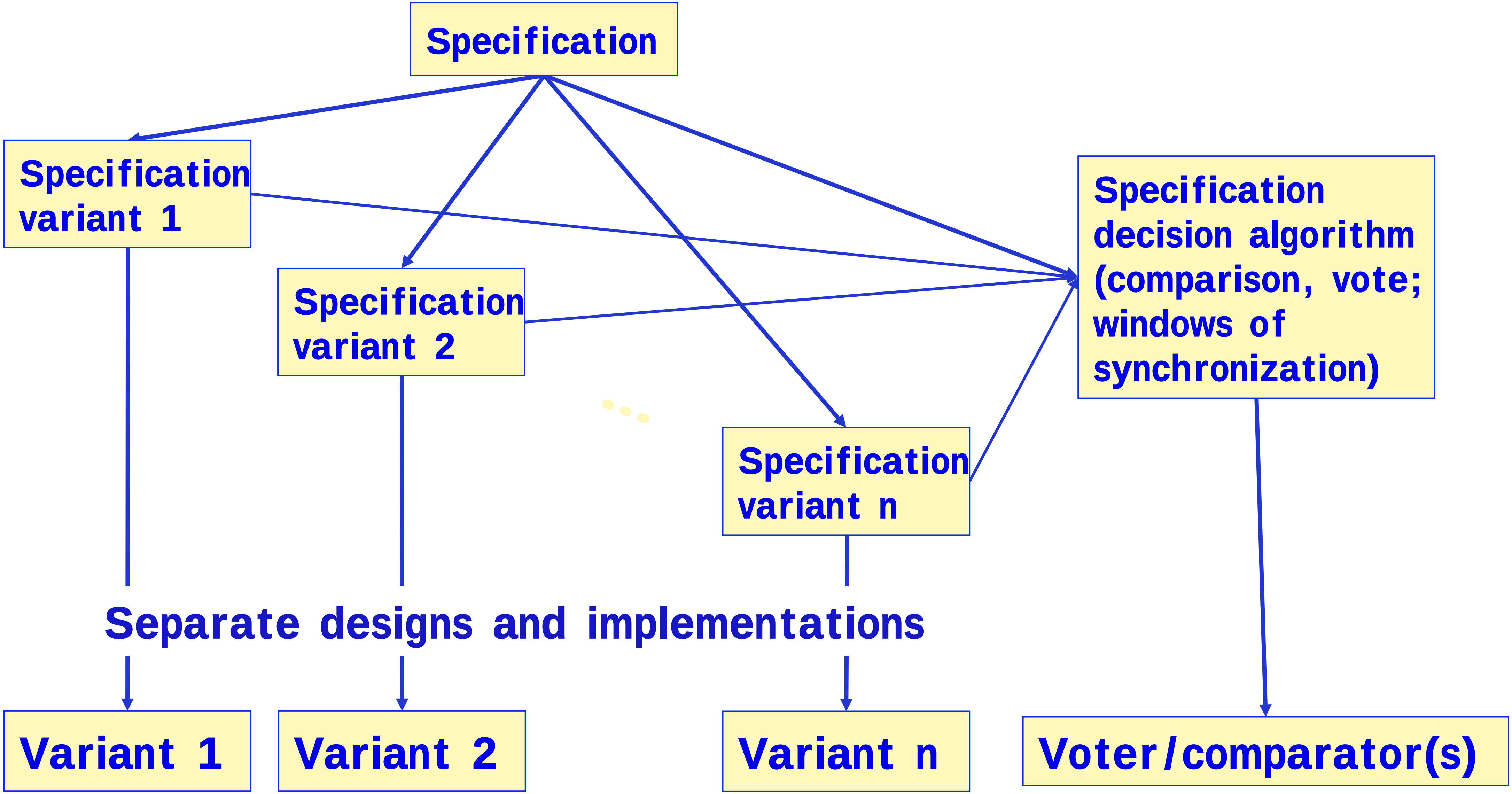


Ci : Si = S ?
Ci = 0 -> Agreement
Ci = 1 -> Discrepancy

	C1	C2	C3	Diagnosis
	0	0	0	No component failed
	1	0	0	Comp. 1 failed
0	1		0	Comp. 2 failed
	0	0	1	Comp. 3 failed
	1	1	1	Voter failed

Reconfiguration after 1st failure?

Development-faults —> Design Diversity



Design Diversity

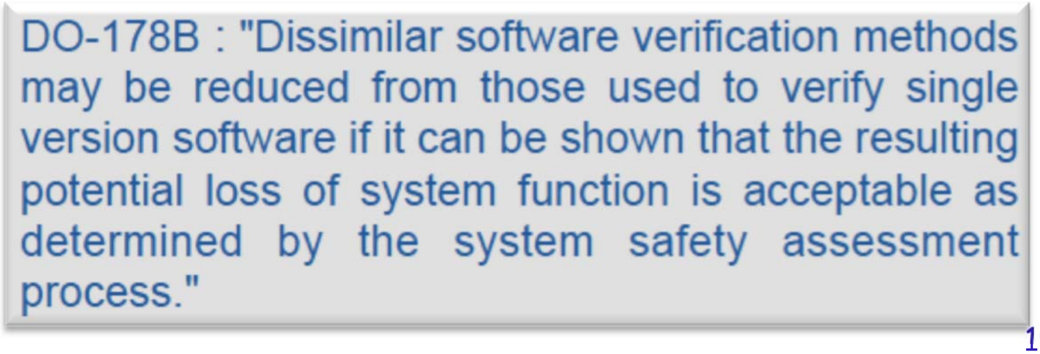
- **Aim: fault independency** (↘ risk of common mode failures)

  Issues: common specification, inter-variant synchronization & decision

- **Major techniques:**
  - Recovery Blocks
  - N-Version Programming
  - N-Self-Checking Prgramming

- **Operational use**
  - Civil aviation: generalized, at differing levels
  - Railway signaling: widely applied
  - Nuclear control: partially used

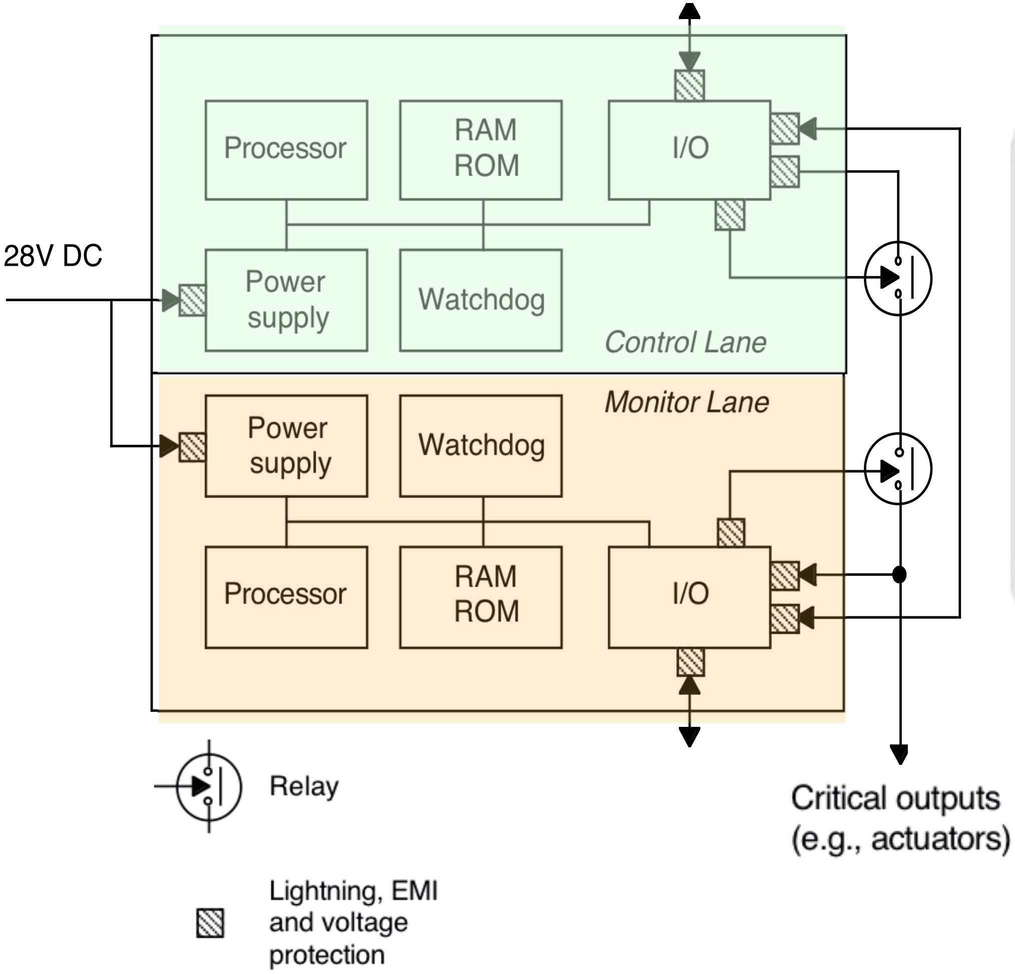
- **Dependability improvement**
  - Real gain for SW faults, although less than wrt HW
  - Verification of specification
  - Impact on Standards 0178-B, IEC 880, CENELEC 50128, IEC 61508, ISO 26262,… —>

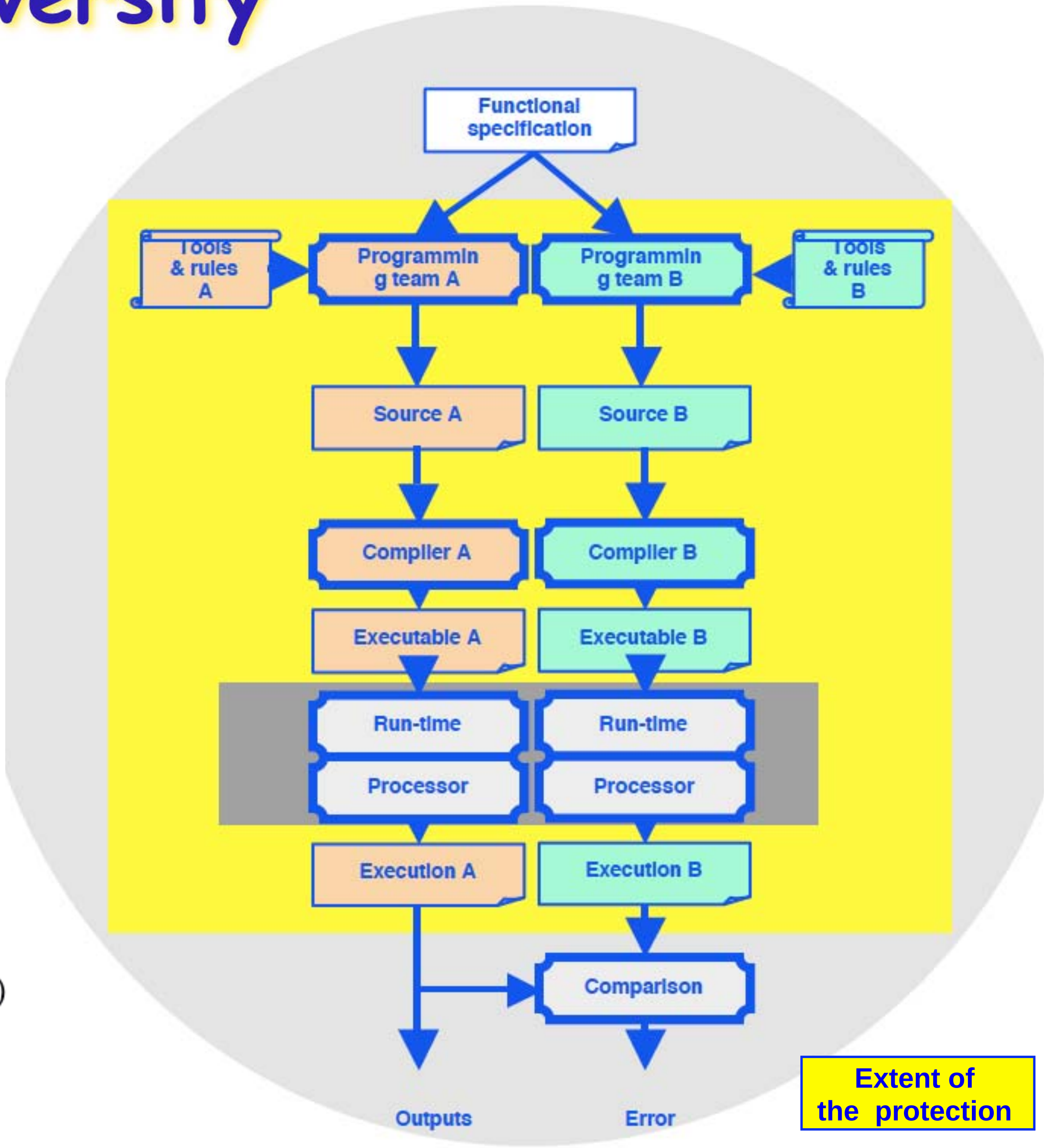


DO-178B : "Dissimilar software verification methods may be reduced from those used to verify single version software if it can be shown that the resulting potential loss of system function is acceptable as determined by the system safety assessment process."

Architectural Principles for Operational Diversity

Airbus A320
(Traverse, Brière 1993)

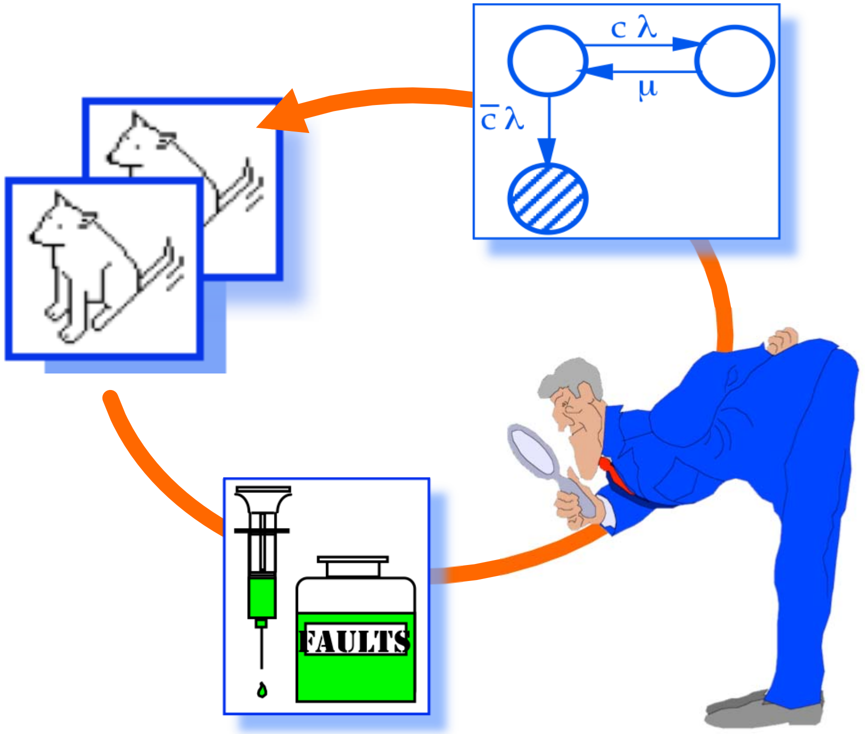




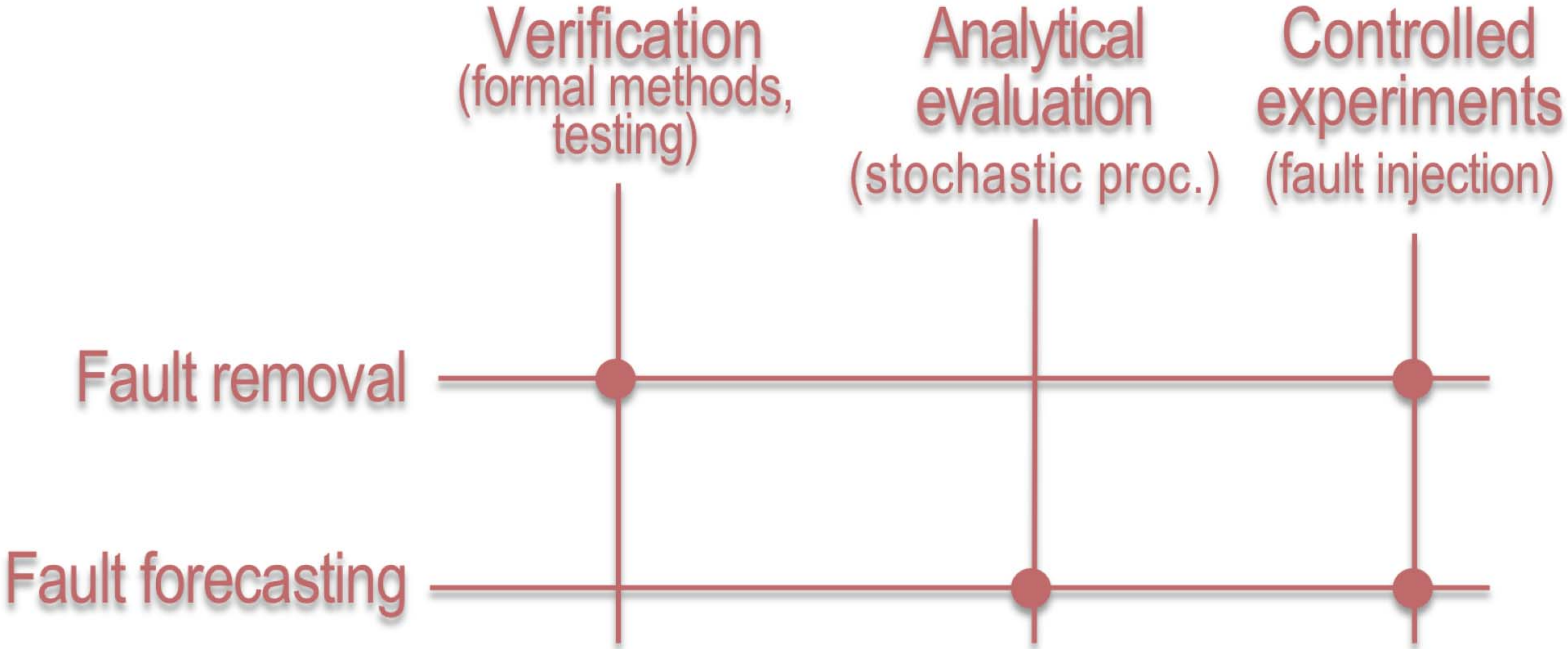
Extent of the protection

Outline

- Dependable Computing
  - Basic Definitions and Terminology
  - Fault Tolerance
- Dependability Assessment
  - Experimental Validation of Fault-Tolerant Computing Systems
  - Dependability Benchmarking of Computers Systems and Components

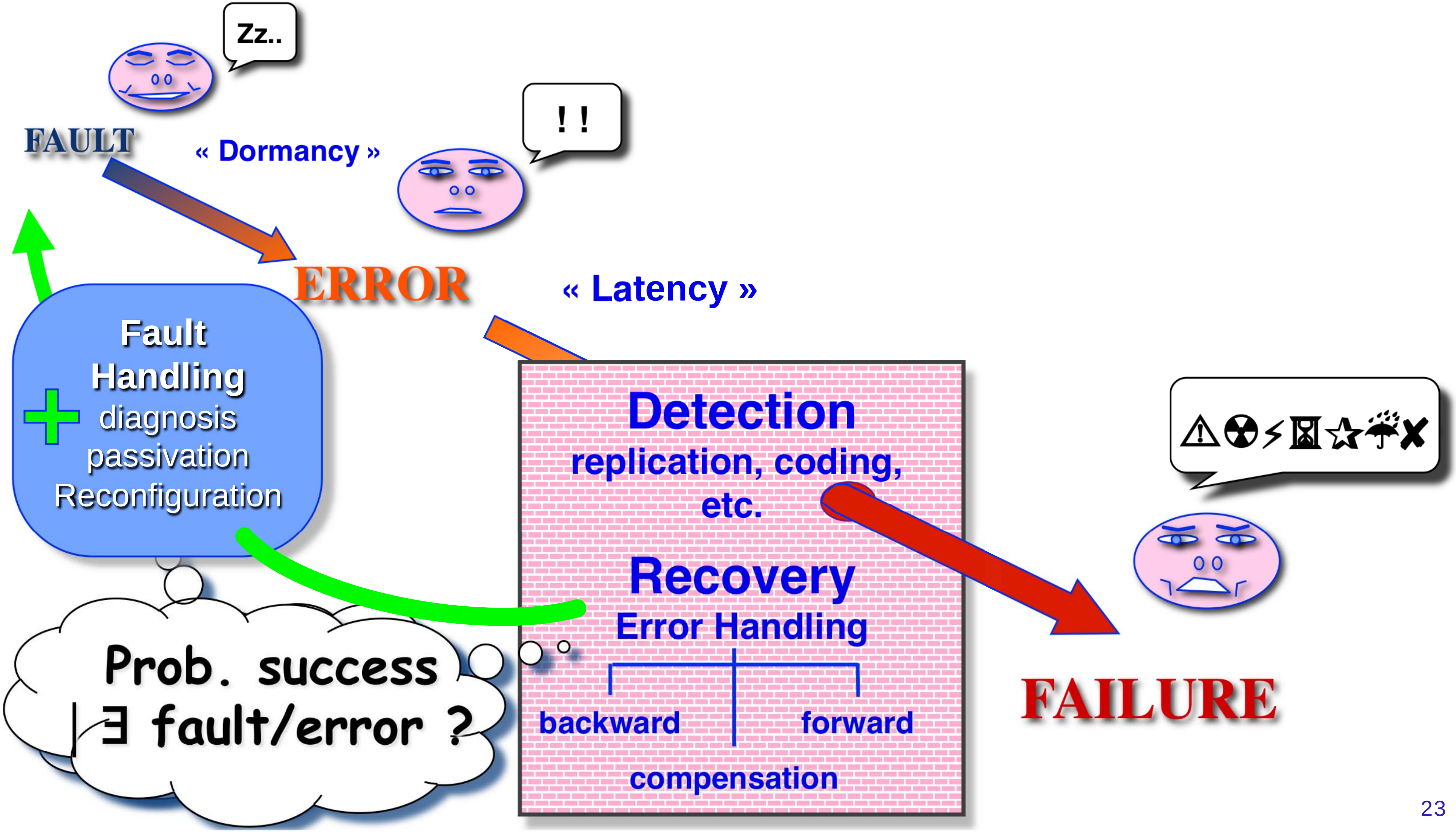


Despendability Assesments Methods



	Verification (formal methods, testing)	Analytical evaluation (stochastic proc.)	Controlled experiments (fault injection)
Fault removal	●		●
Fault forecasting		●	●

Fault Tolerance … and Coverage



FAULT

« Dormancy »

ERROR

« Latency »

FAILURE

Fault Handling
diagnosis
passivation
Reconfiguration

Detection
replication, coding, etc.

Recovery
Error Handling

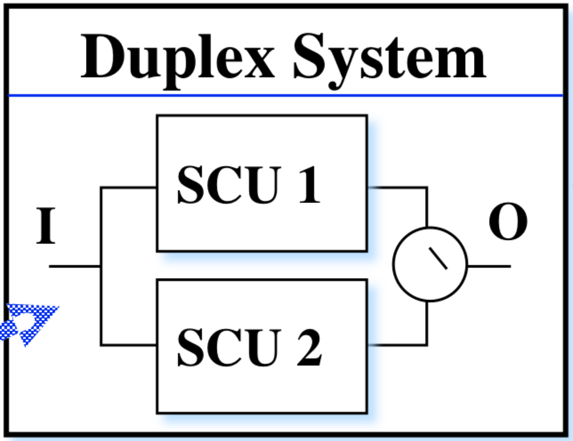
backward

forward

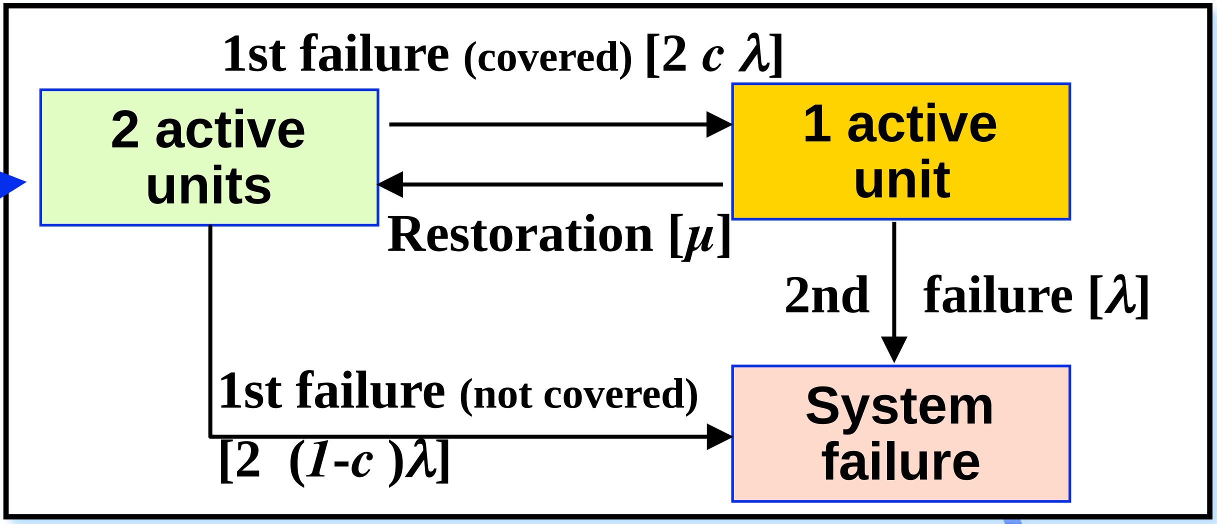
compensation

Prob. success
| ∃ fault/error ?

Impact of Coverage on Dependability



Duplex System



2 active units

1 active unit

System failure

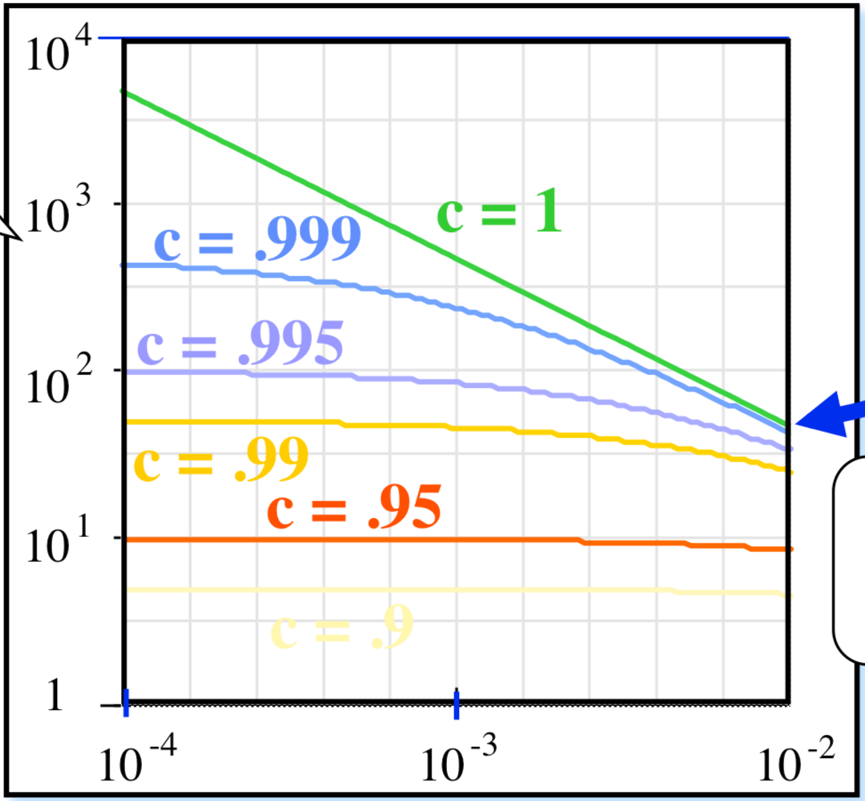
1st failure (covered) [2 *c* λ]

Restoration [μ]

2nd failure [λ]

1st failure (not covered) [2 (*1*-*c*)λ]

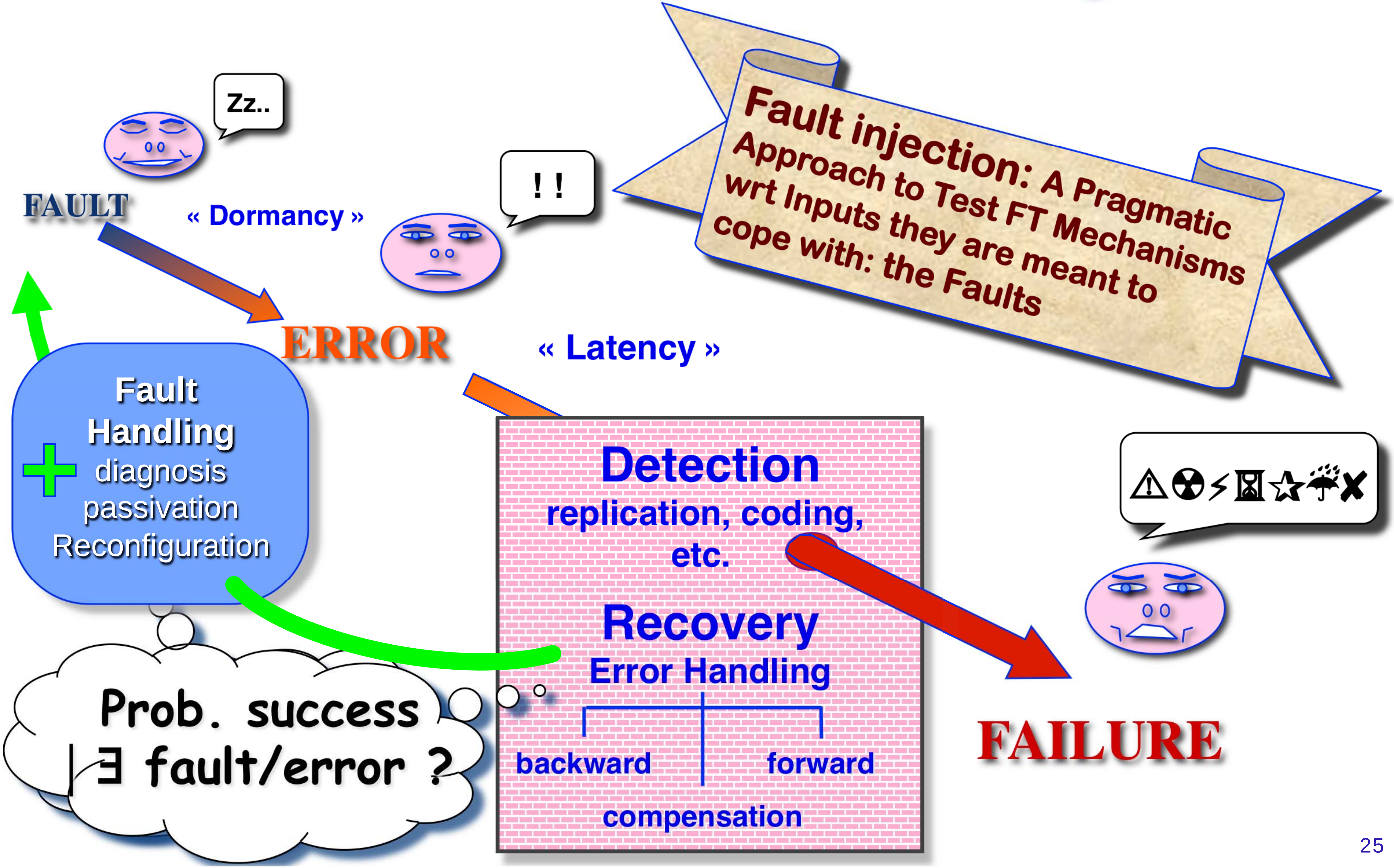
$$\frac{\mathrm{MTTF}_{\mathrm{Syst.}}}{\mathrm{MTTF}_{\mathrm{Unit.}}}$$



$$\frac{\mathrm{MTTR}_{\mathrm{Comp.}}}{\mathrm{MTTF}_{\mathrm{Comp.}}} \left(\frac{\lambda}{\mu}\right)$$

J. Arlat, A. Costes, Y. Crouzet, J.-C. Laprie,
D. Powell
Fault Injection and Dependability Evaluation
of Fault-Tolerant Systems
IEEE ToC, 42 , (8), pp. 913 – 923, August 1993

Fault Tolerance … and Coverage



Zz..

FAULT

« Dormancy »

! !

ERROR

« Latency »

Fault injection: A Pragmatic Approach to Test FT Mechanisms wrt Inputs they are meant to cope with: the Faults

Fault Handling
+ diagnosis
passivation
Reconfiguration

Detection
replication, coding, etc.

Recovery
Error Handling

backward | forward

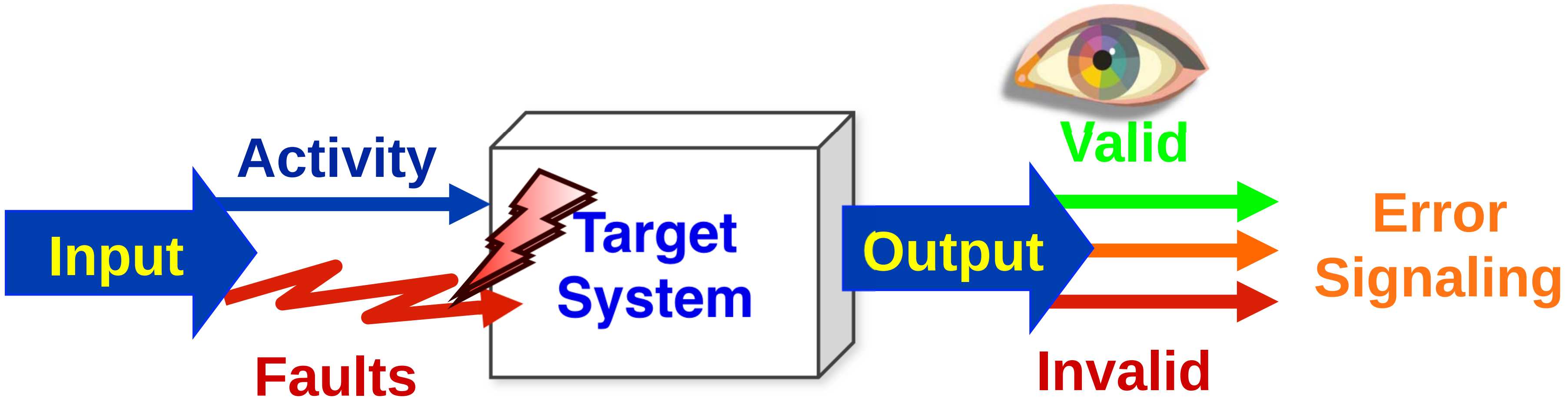
compensation

⚠☢⚡⌛☆☔✗

FAILURE

Prob. success |∃ fault/error ?

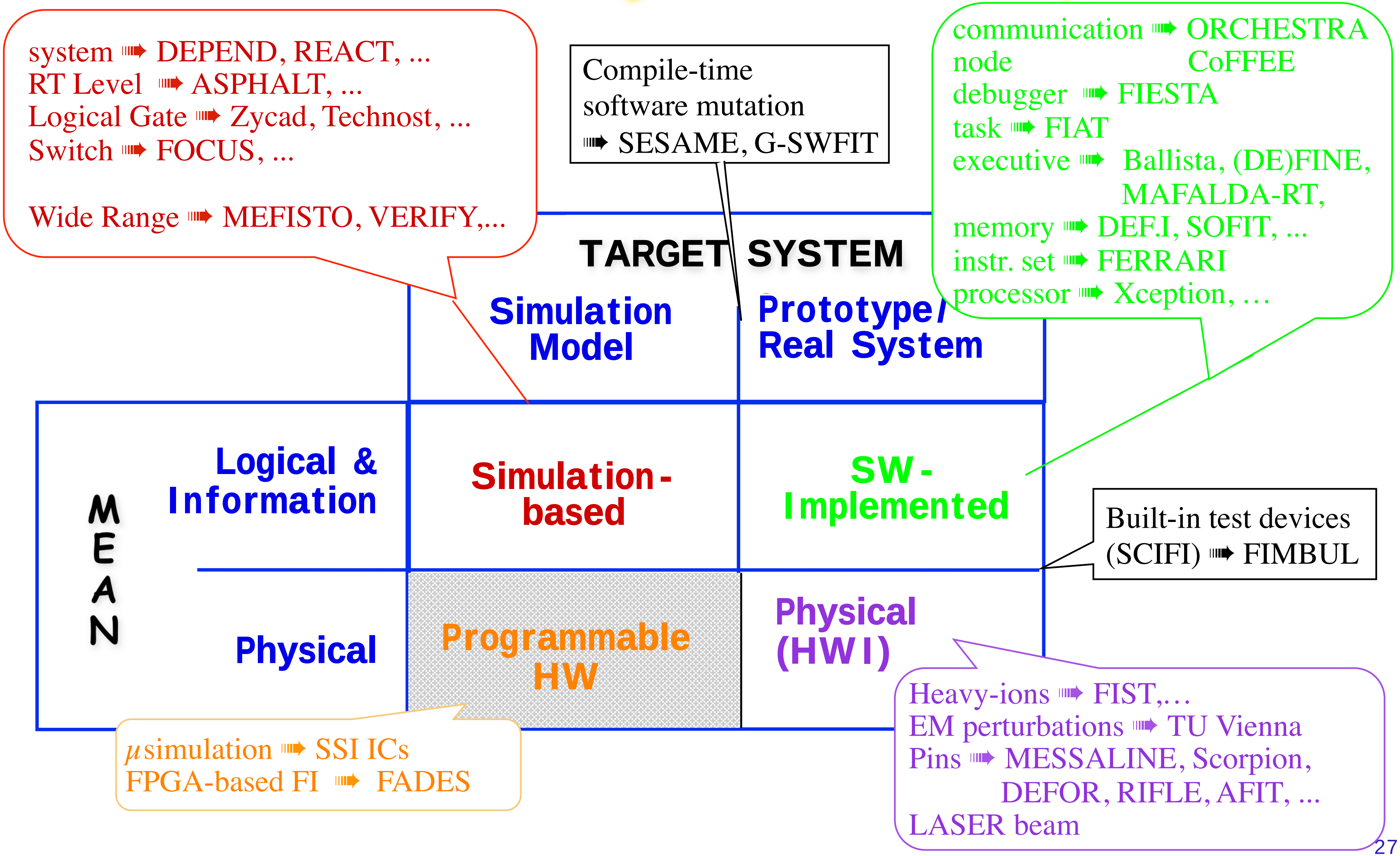
Fault Injection-based Assessment



—> Partial dependability assessment: controlled application of fault/error conditions

- Testing and evaluation of a fault-tolerant computer system and of its FT algorithms & mechanisms
- Characterization of faulty behaviors & failure modes of several computer systems & components
  -> Dependability benchmarking (comparison purpose)

The Fault Injection Techniques



		TARGET SYSTEM	
		Simulation Model	Prototype / Real System
MEAN	Logical & Information	Simulation-based	SW-Implemented
	Physical	Programmable HW	Physical (HWI)

system ➠ DEPEND, REACT, ...
RT Level ➠ ASPHALT, ...
Logical Gate ➠ Zycad, Technost, ...
Switch ➠ FOCUS, ...

Wide Range ➠ MEFISTO, VERIFY,...

Compile-time
software mutation
➠ SESAME, G-SWFIT

communication ➠ ORCHESTRA
node CoFFEE
debugger ➠ FIESTA
task ➠ FIAT
executive ➠ Ballista, (DE)FINE,
MAFALDA-RT,
memory ➠ DEF.I, SOFIT, ...
instr. set ➠ FERRARI
processor ➠ Xception, …

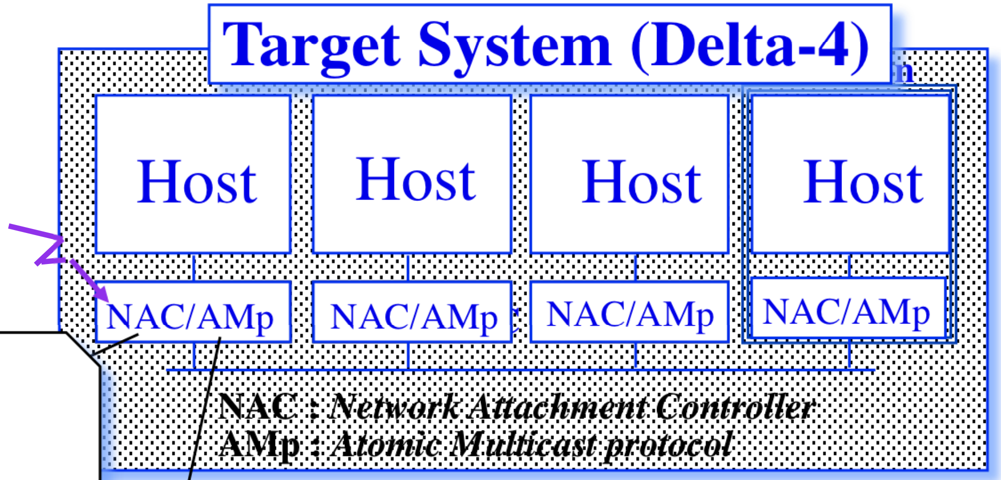
Built-in test devices
(SCIFI) ➠ FIMBUL

μsimulation ➠ SSI ICs
FPGA-based FI ➠ FADES

Heavy-ions ➠ FIST,…
EM perturbations ➠ TU Vienna
Pins ➠ MESSALINE, Scorpion,
DEFOR, RIFLE, AFIT, ...
LASER beam

Examples of Experimental Results - 1

Physical Injection (MESSALINE)



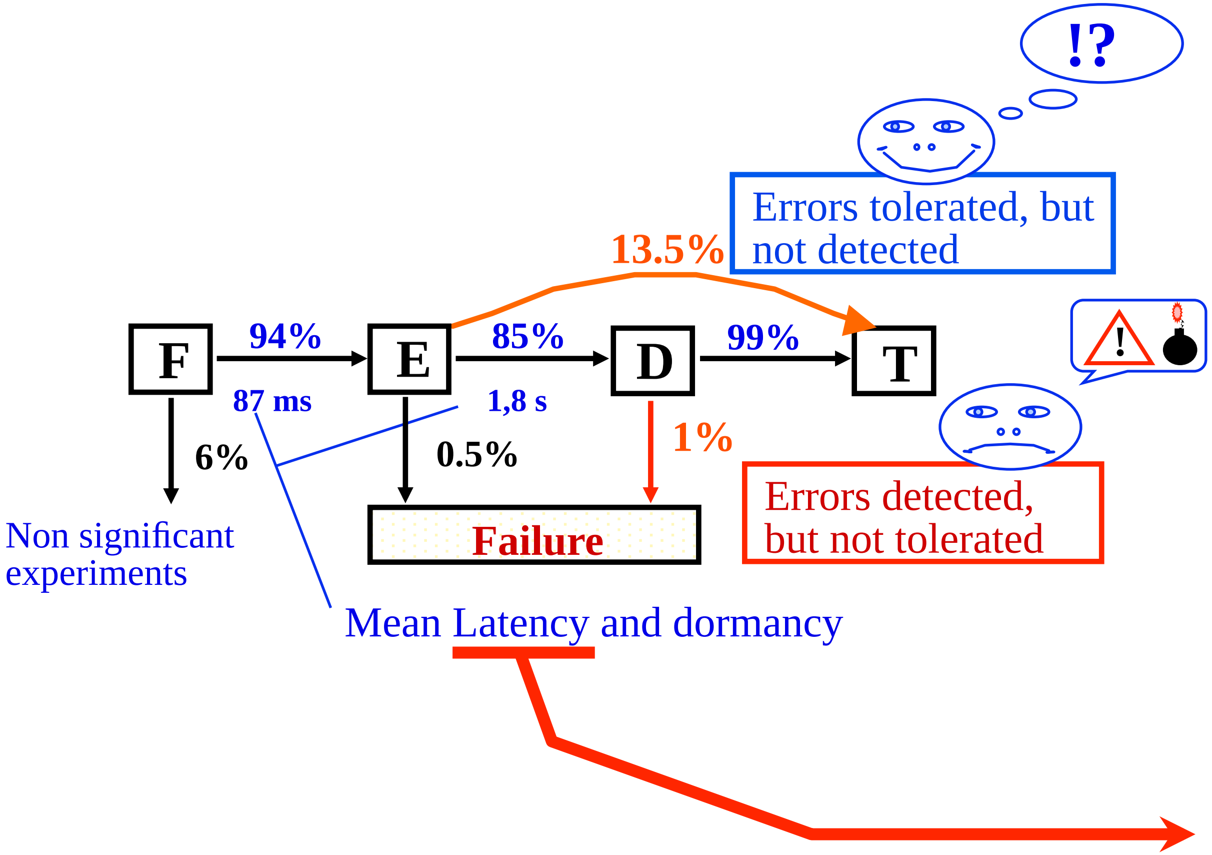
1) "Standard"
[limited self-checking]
2) Duplex Arch.
[enhanced self-checking]

Several releases ofAMp

T Predicate:
Protocol properties OK and error confined

D Predicate:
Self extraction of the injected station

Examples of Experimental Results - 2



!?

Errors tolerated, but not detected

13.5%

F

94%

E

85%

D

99%

T

87 ms

1,8 s

6%

0.5%

1%

Non significant experiments

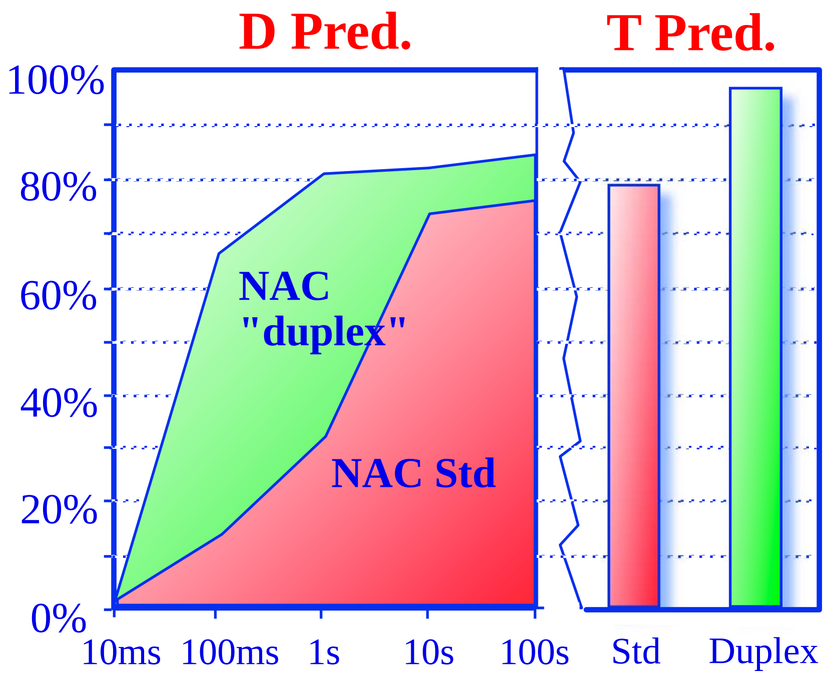
Failure

Errors detected, but not tolerated

Mean Latency and dormancy

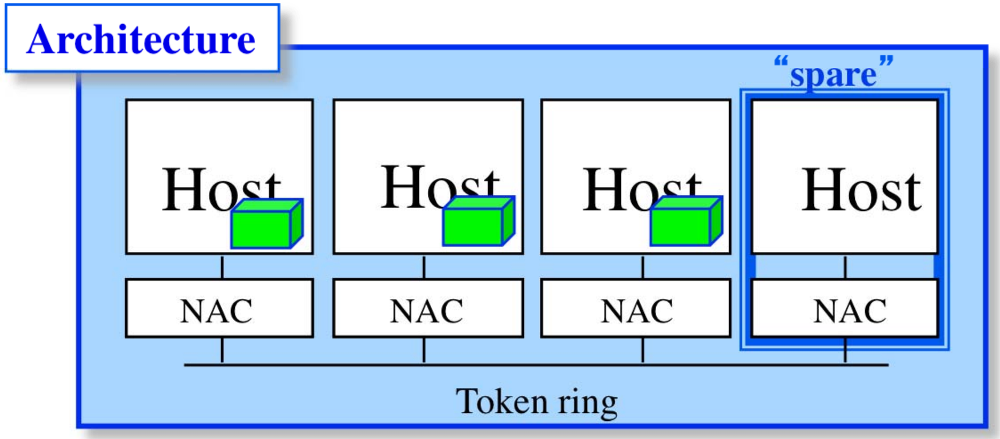
T Predicate:
Protocol properties OK
and error confined

D Predicate:
Self extraction
of the injected station



Link between Exp. & Anal. Eval.: An Example

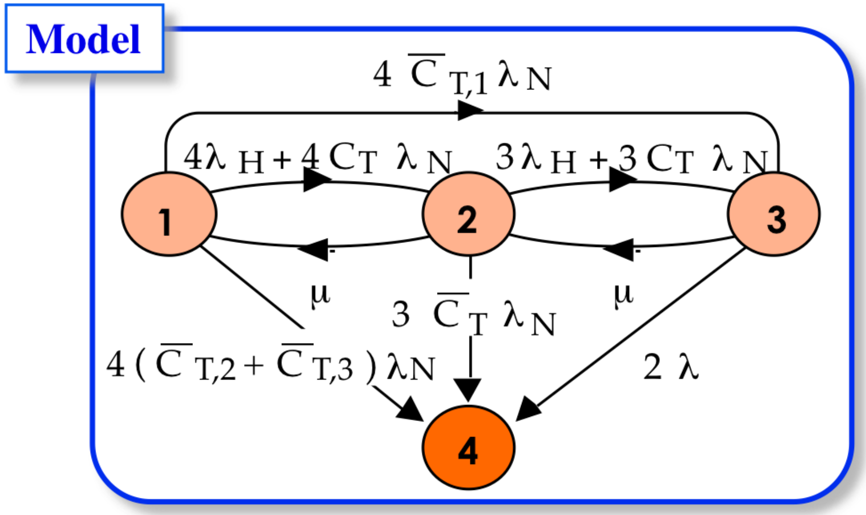
Architecture



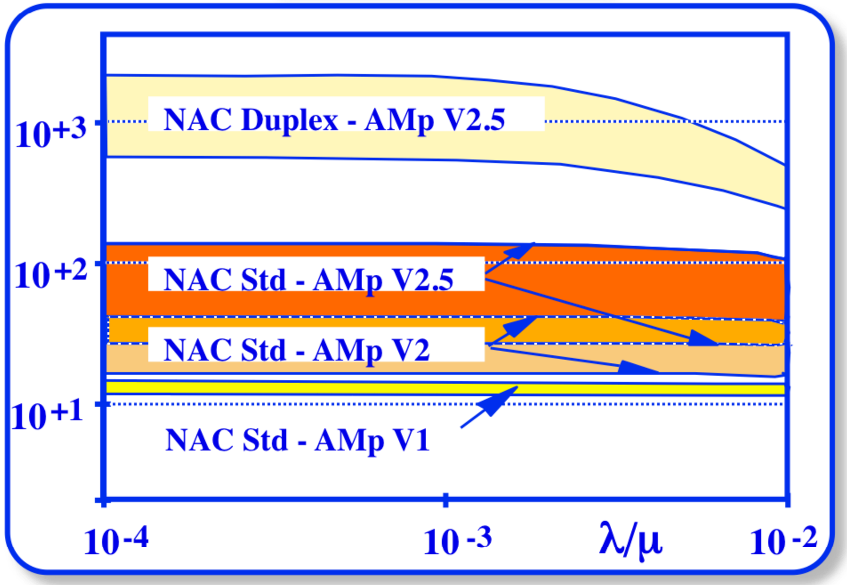
Coverage Factors

Target System	C_T	$\overline{C}_{T,1}$	$\overline{C}_{T,2}$	$\overline{C}_{T,3}$
NAC Std - AMp V 1	79,08%	2,32%	11,77%	6,83%
[illegible] V 2	[illegible]	8,73%	2,8[illegible]	[illegible]45%
N[illegible]td -AMp V [illegible]	[illegible],32%	7,79%	1,0[illegible]	[illegible]
NAC Duplex - AMp V 2.5	99,55%	0,32%	0,00%	0,12%

Model

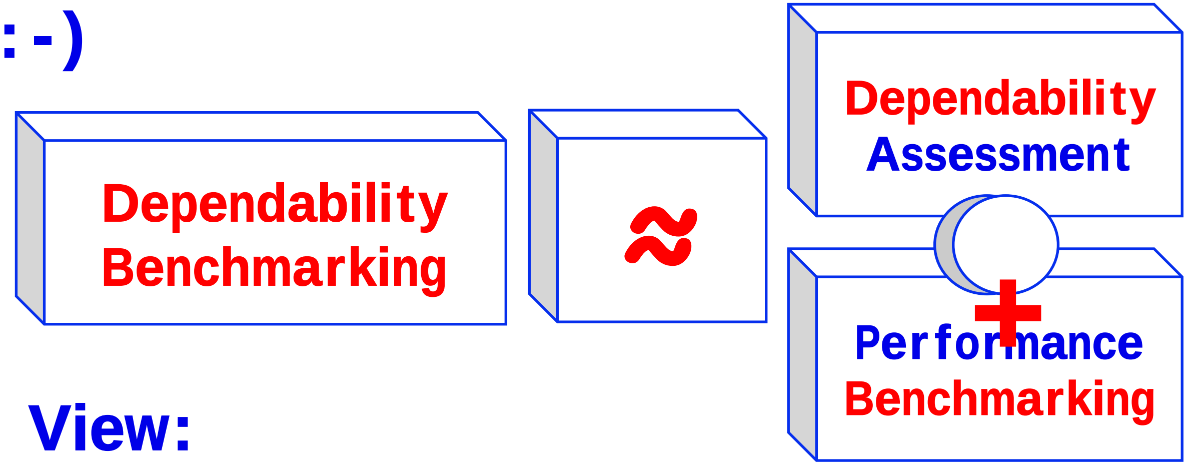

$4\ \overline{C}_{T,1}\ \lambda_N$
$4\lambda_H + 4\,C_T\ \lambda_N$
$3\lambda_H + 3\,C_T\ \lambda_N$
1
2
3
4
μ
μ
$3\ \overline{C}_T\ \lambda_N$
$4\,(\overline{C}_{T,2} + \overline{C}_{T,3})\,\lambda_N$
$2\ \lambda$

MTFF network
MTFF station



Views about Dependability Benchmarking

Naive View … :-)



Dependability Benchmarking

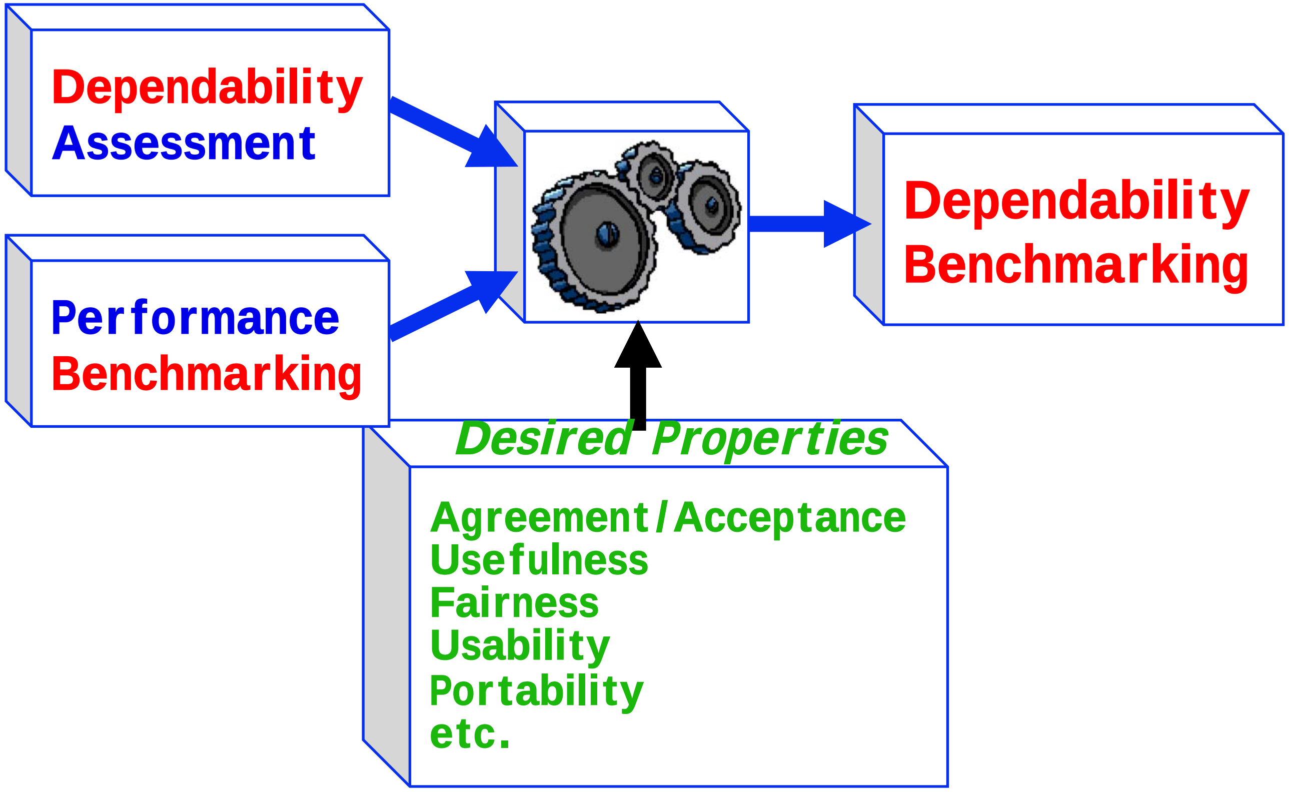
≈

Dependability Assessment

+

Performance Benchmarking

More Realistic View:



Dependability Assessment

Performance Benchmarking

Dependability Benchmarking

Desired Properties

Agreement/Acceptance
Usefulness
Fairness
Usability
Portability
etc.

FI Campaign *vs.* Dependability Benchmark

FTS Assessment

- 1 Target System
- In-Deep Knowledge OK
- FTMs testing
- Fault and Activity sets
- Sophisticated faults
- Measures = conditional dependability assessment
- One-of-a-kind process: “heavy duty” still OK
- Developer’s view
- Results published, experiment context often proprietary

Dependability Benchmarking

- > 1 Target Systems [Components]
- Limited Knowledge only
- Global system behavior
- Fault- and Work-load
- Reference (interface) faults
- Measures = Dependability assess. —> Fault occurrence process
- Recurring process: “user friendly” required
- End User/Integrator’s view
- Results and procédure openly disclosed

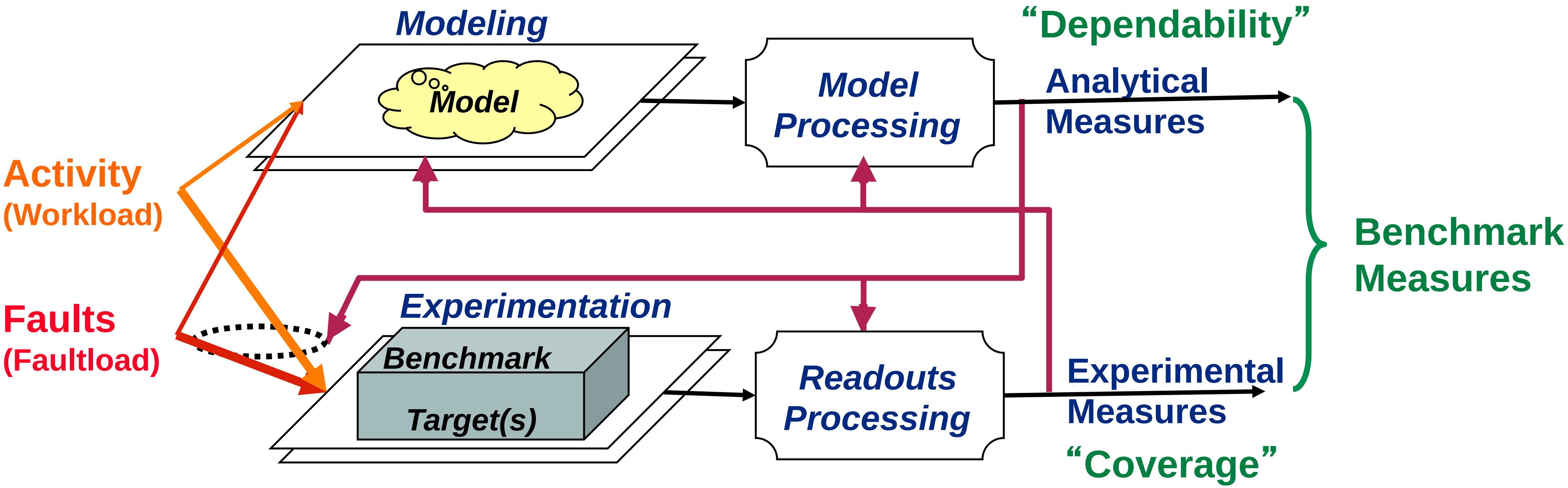
Common Properties

Non Intrusiveness: No influence on temporal behavior, nor target system alteration

Representativeness: Fault and Activity/Work set/loads

Repeatability: Derivation of statistically equivalent results

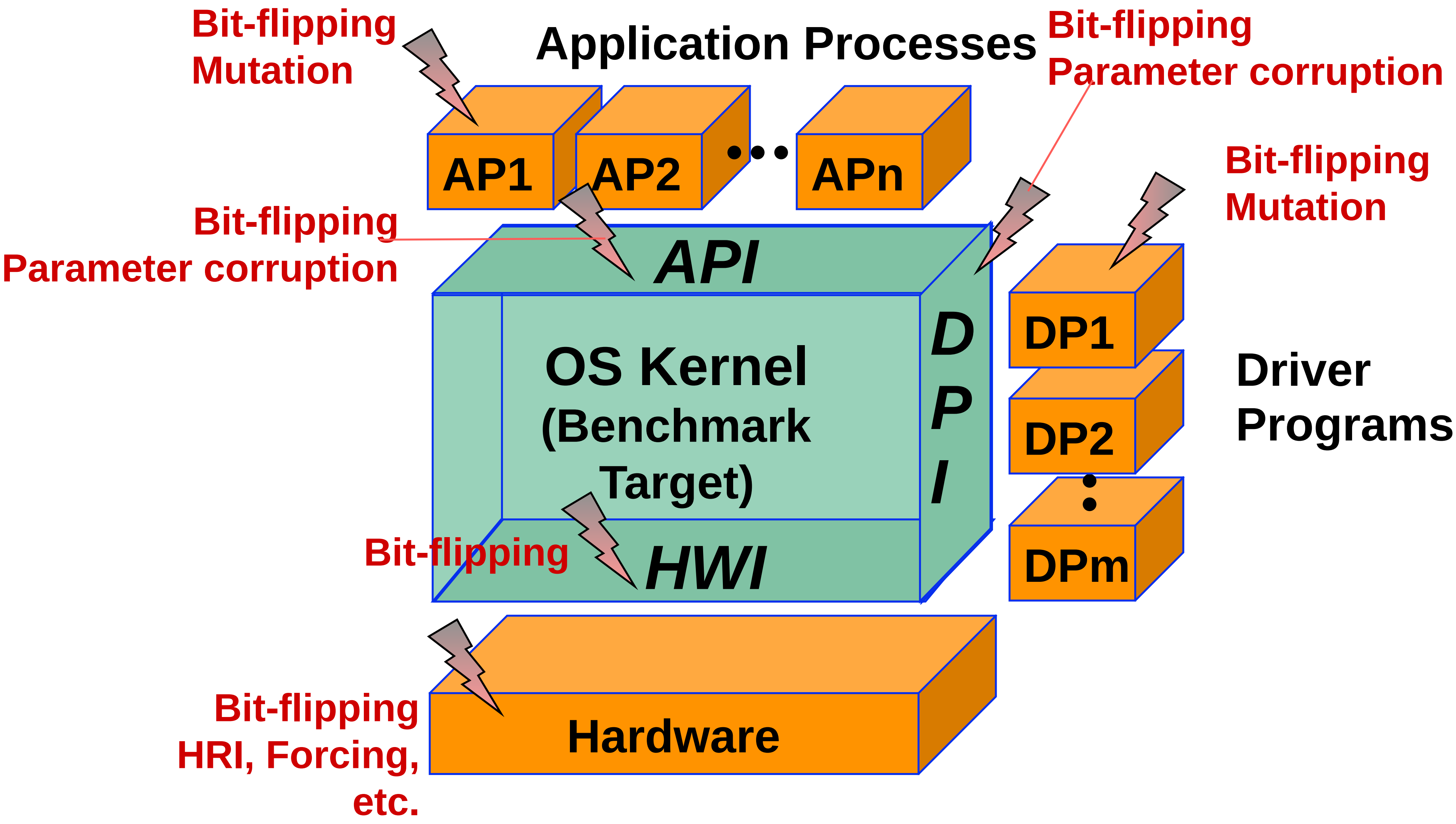
A Comprehensive Dependability Assessment Frame



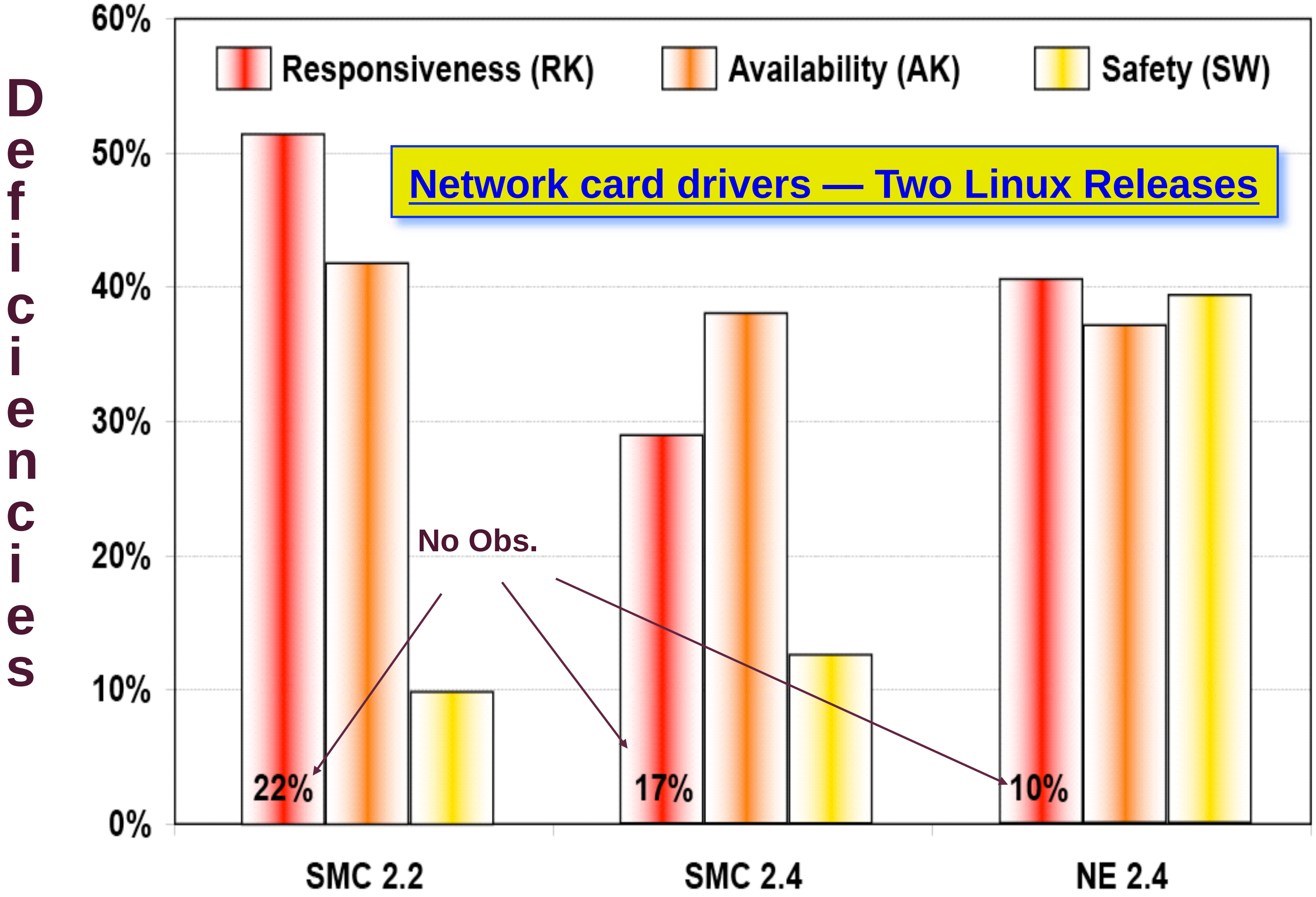
IST Project DBench (*Dependability Benchmarking*) — www.laas.fr/DBench and www.dbench.org

—> Minimal set of data needed from the Target System(s) (architecture, configuration, operation, environment, etc.) to derive actual dependability attributes?

About Interfaces (SW Executive)



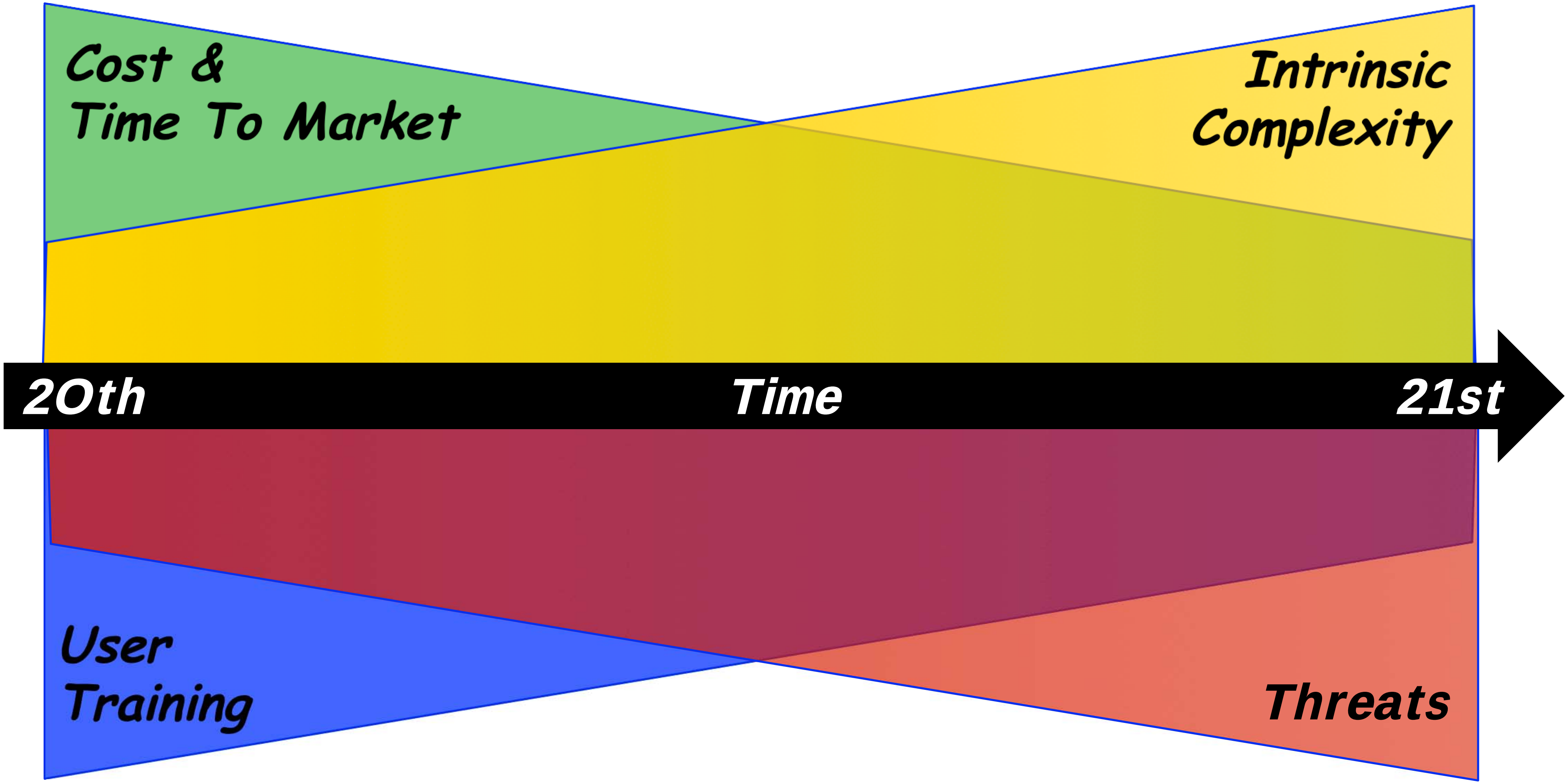
Impact of Peripheral Drivers & Dependability Viewpoints



System call parameter corruption at DPI

A. Albinet, J. Arlat, J.-C. Fabre
Benchmarking the Impact of Faulty Drivers: Application to the Linux Kernel
Dependability Benchmarking for Computer Systems (K. Kanoun, L. Spainhower, Eds.), pp. 285-310, 2008

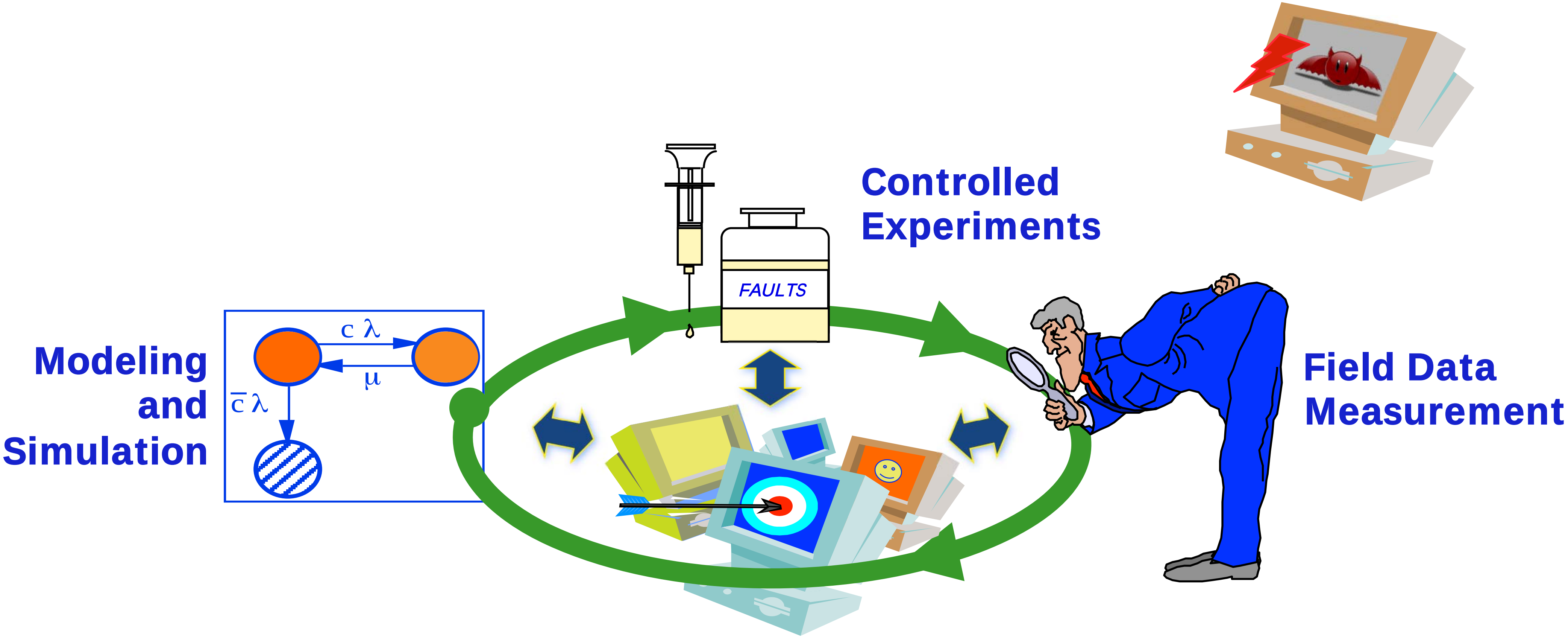
Looking Ahead: An Ever Moving Target



See also:
D. Siewiorek, R. Chillarege, Z. Kalbarczyk
Reflections on Industry Trends and Experimental Research in Dependability
IEEE TDSC, Vol. 1, No. 2, April-june 2004, pp. 109-127

Comprehensive Assessment Framework

Emerging Features and Challenges

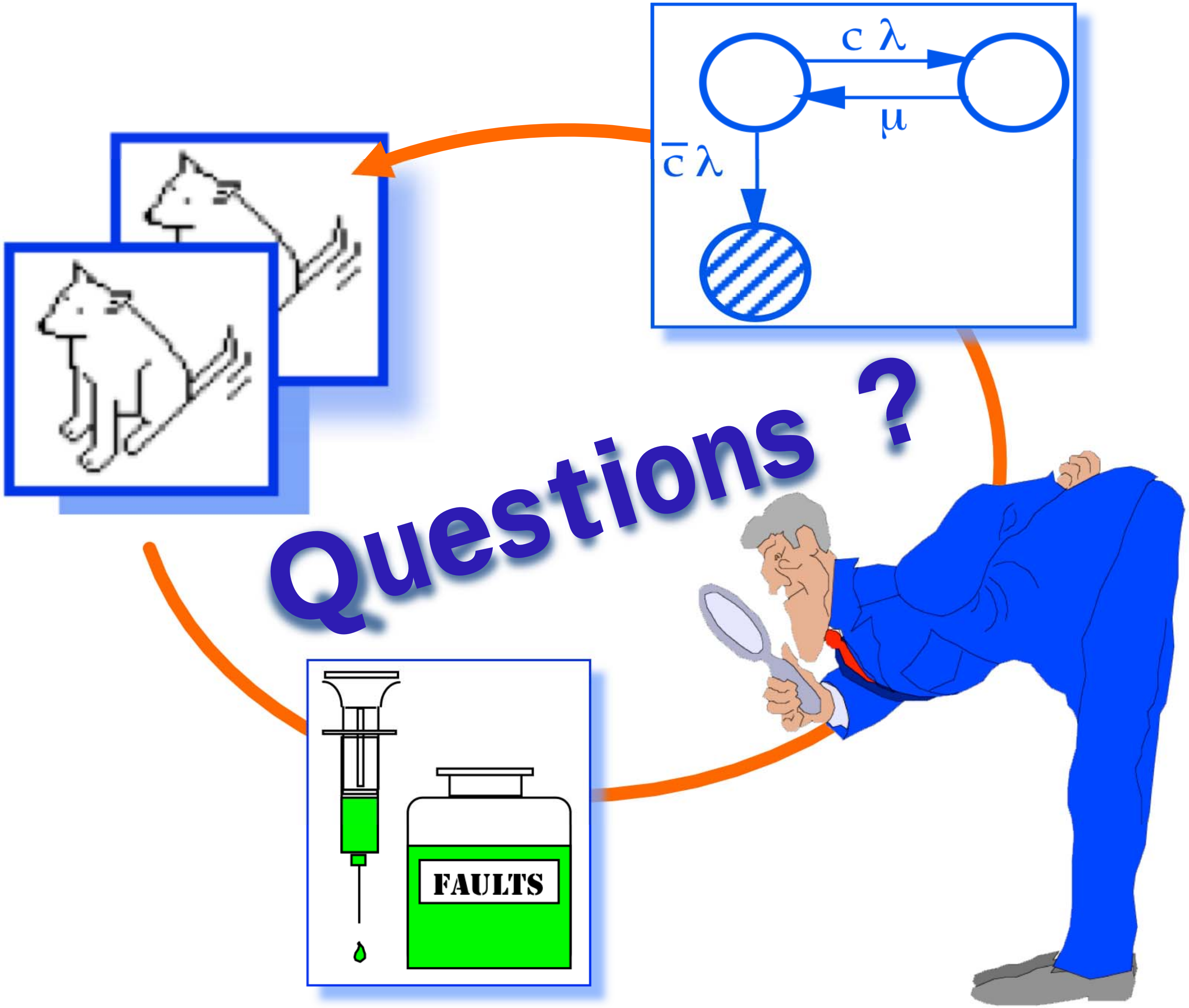


Modeling and Simulation

Controlled Experiments

Field Data Measurement

Mobility Configurability Target System … Highly evolutive Attacks



Questions ?