International France-China Workshop

New and Smart Information Communication Science and Technology to support Sustainable Development (NICST 2013)

September 18-20, 2013 — Clermont-Ferrand, France

Towards Dependable Computing: The Self-reinforcing Architecting and Assessment Loop

Jean Arlat

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

LAAS-CNRS

INSTITUT CARNOT LAAS CNRS

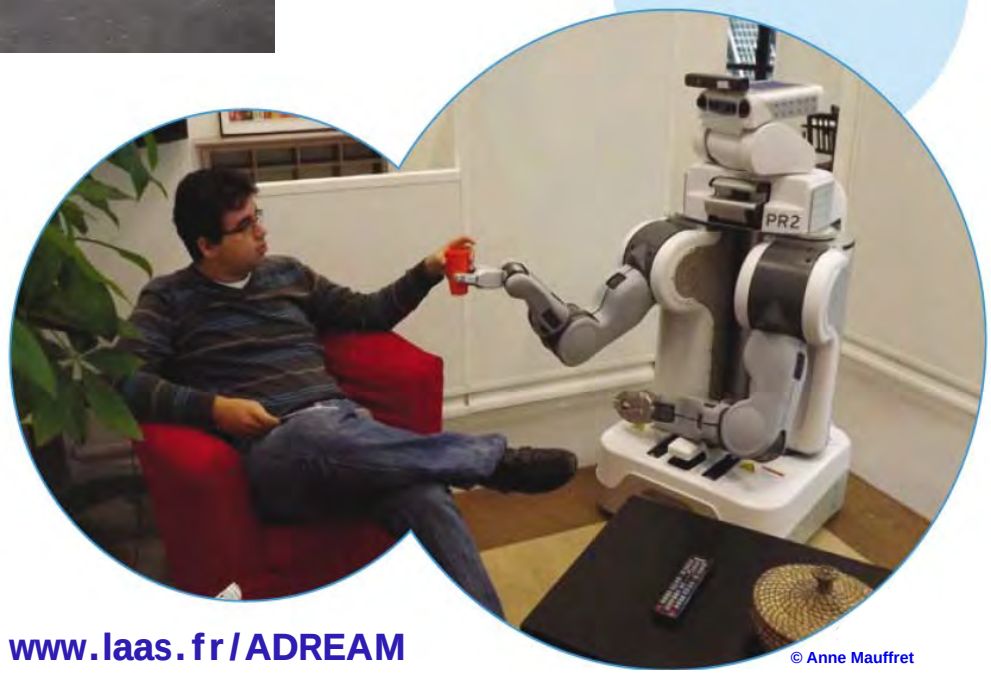
www.laas.fr

Université de Toulouse

Tomorrow's is (almost) Here Today
Some Perspectives

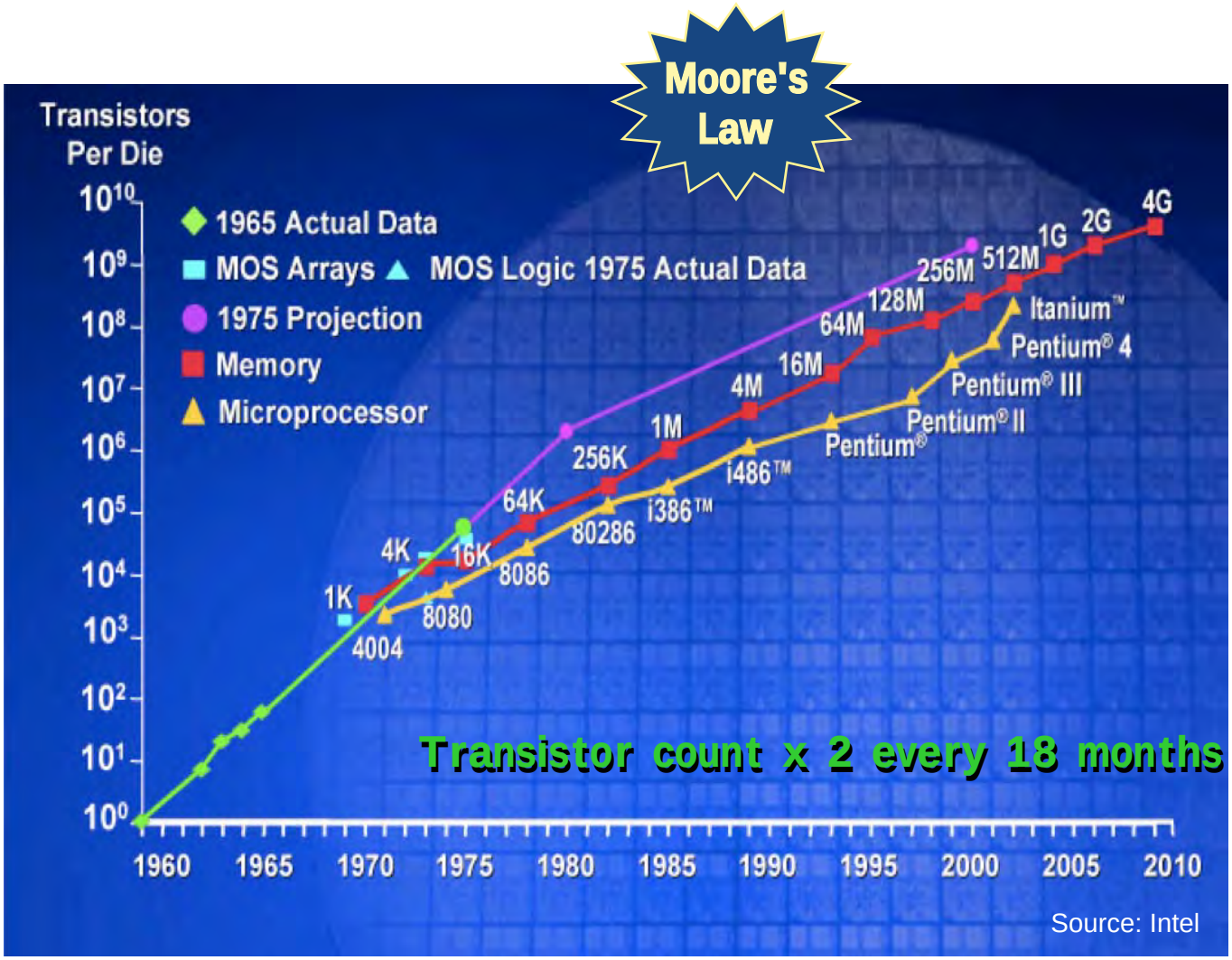
- **Emerging Services and Trends**
  - Guidance in Public space
  - Assistance to Elderly people,…
  - *Unmanned* search, Rescue and Recovery
  - Smart Grids for Heterogeneous and Distributed "supply chain": control, monitoring and metering
  - Car *and* Home Energy Management
  - Autonomous Individual Vehicles Systems, On-demand transportation
  - Factory of the Future (Workshop with Humans and Robot Co-workers)
- **Integration of Information Processing into Everyday Objects and Activities**
  - Harware and Software Technologies Development
  - Interconnection and Communication Capabilities
  - Internet of Things, Ambient Intelligence, Cyber-physical Systems, …

The ADREAM PLatform @ LAAS-CNRS



www.laas.fr/ADREAM

Trend in Hardware Technology



- Performance ↗
- Clock frequency ↗

-> An ever growing set of smarter services
=> Emergence of cyber-physical systems

But:

- Power dissipation ↗
- Process variations ↗
- Manufacturing costs ↗
- Yield ↘
- Prob. Defects undetected ↗
- "Soft" Error Rate ↗

"Less than Perfect" Circuits (Manufacturing Defects and Transient Faults)

—> Resilience Achieved via Redundancy Techniques

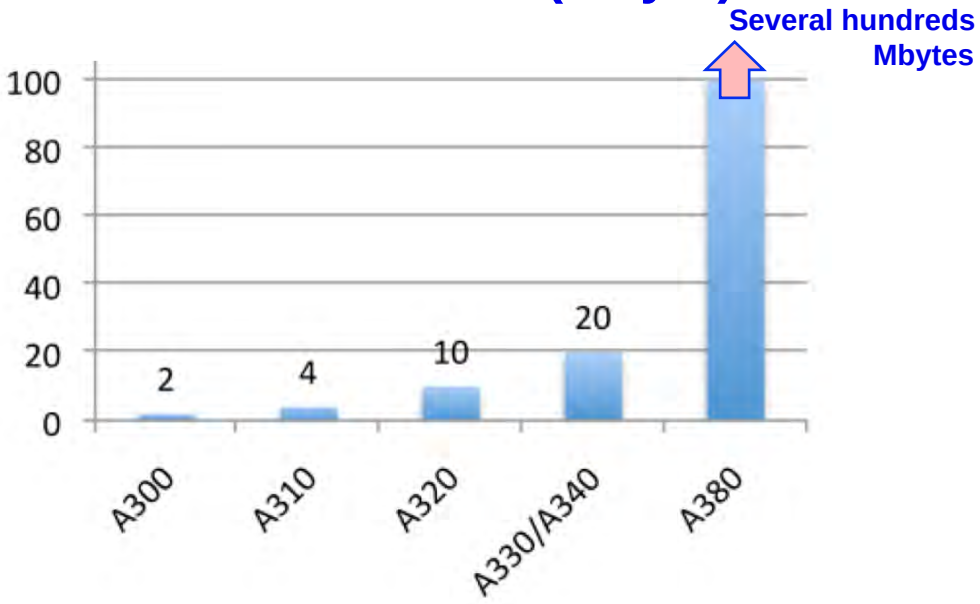
International Technology Roadmap for Semiconductors

- **Crosscutting Challenge 5: Reliability (2008 Update)**
  Reliability & Resilience (2009 Edition)
- **2011 Edition/ 2012 Update: Design for Reliability and Resilience confirmed as "new long-term *Grand Challenge*"**
  (together with design of concurrent software)
  "Design Technology for Resilience: A Fundamental Portion of DFM"
- **Quoting the Design Section** [http://www.itrs.net/Links/2011ITRS/2011Chapters/2011Design.pdf]
  - ***Relaxing the requirement of 100% correctness for devices and interconnects may dramatically reduce costs of manufacturing, verification, and test***
  - ***Such a paradigm shift will likely be forced in any case by technology scaling, which leads to more transient and permanent failures of signals, logic values, devices, and interconnects***
  - ***In general, automatic insertion of robustness into the design will become a priority as systems become too large to be functionally tested at manufacturing exit***
  - ***Potential solutions include automatic introduction of redundant logic and on-chip reconfigurability for fault tolerance, development of adaptive and self-correcting or self-healing circuits, and software-based fault-tolerance***

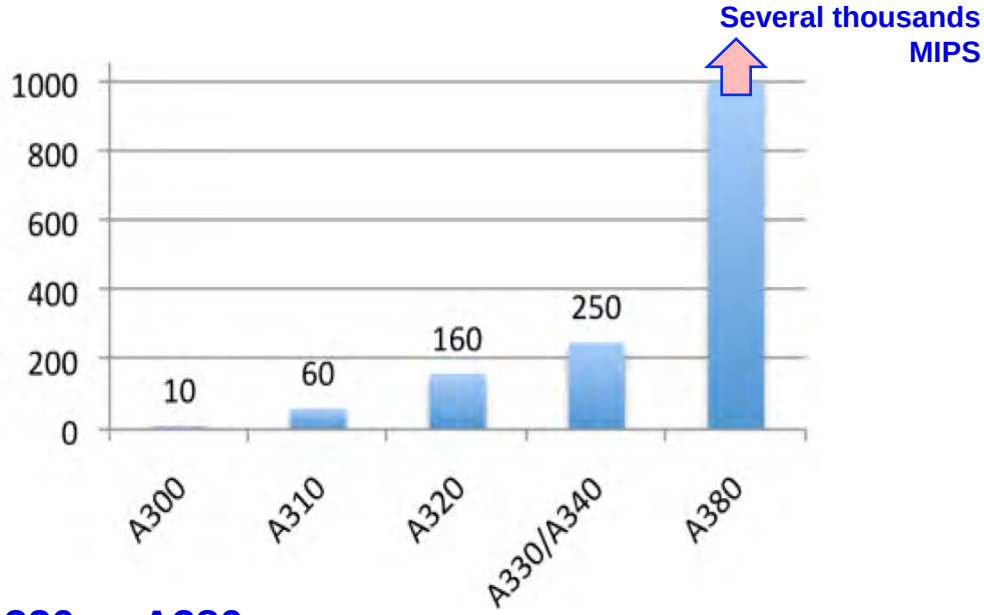
Increased Functionalities and Complexity of Transportation Systems

■ Current Civil Aircraft

Size of Software (Mbyte)



Computation Power (MIPS)

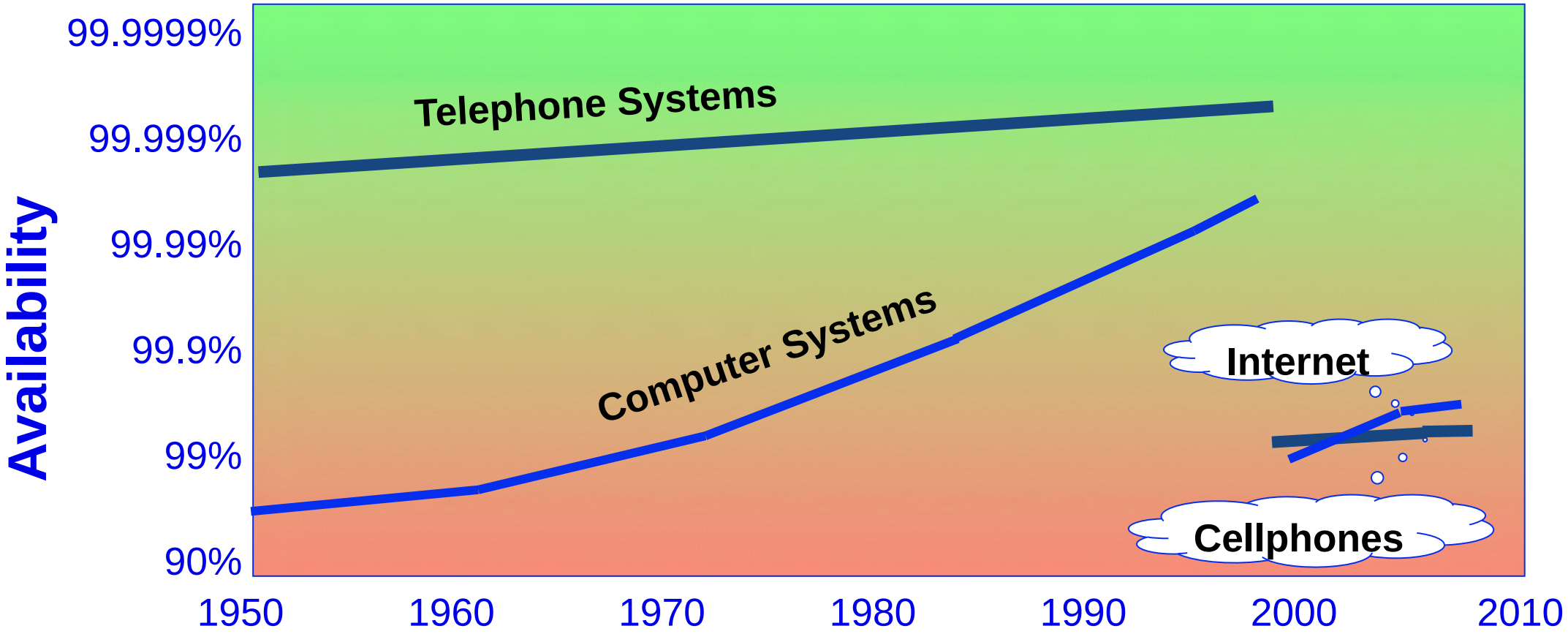


Aircraft	A320	A380
# messages exchanged among embedded systems	2,000	> 100,000

■ Automotive

- Cost of “electronics” in a vehicule > 30% in 2010
- SW code size: several 10’s of Mbytes by this decade

Evolution of Information Infrastructures



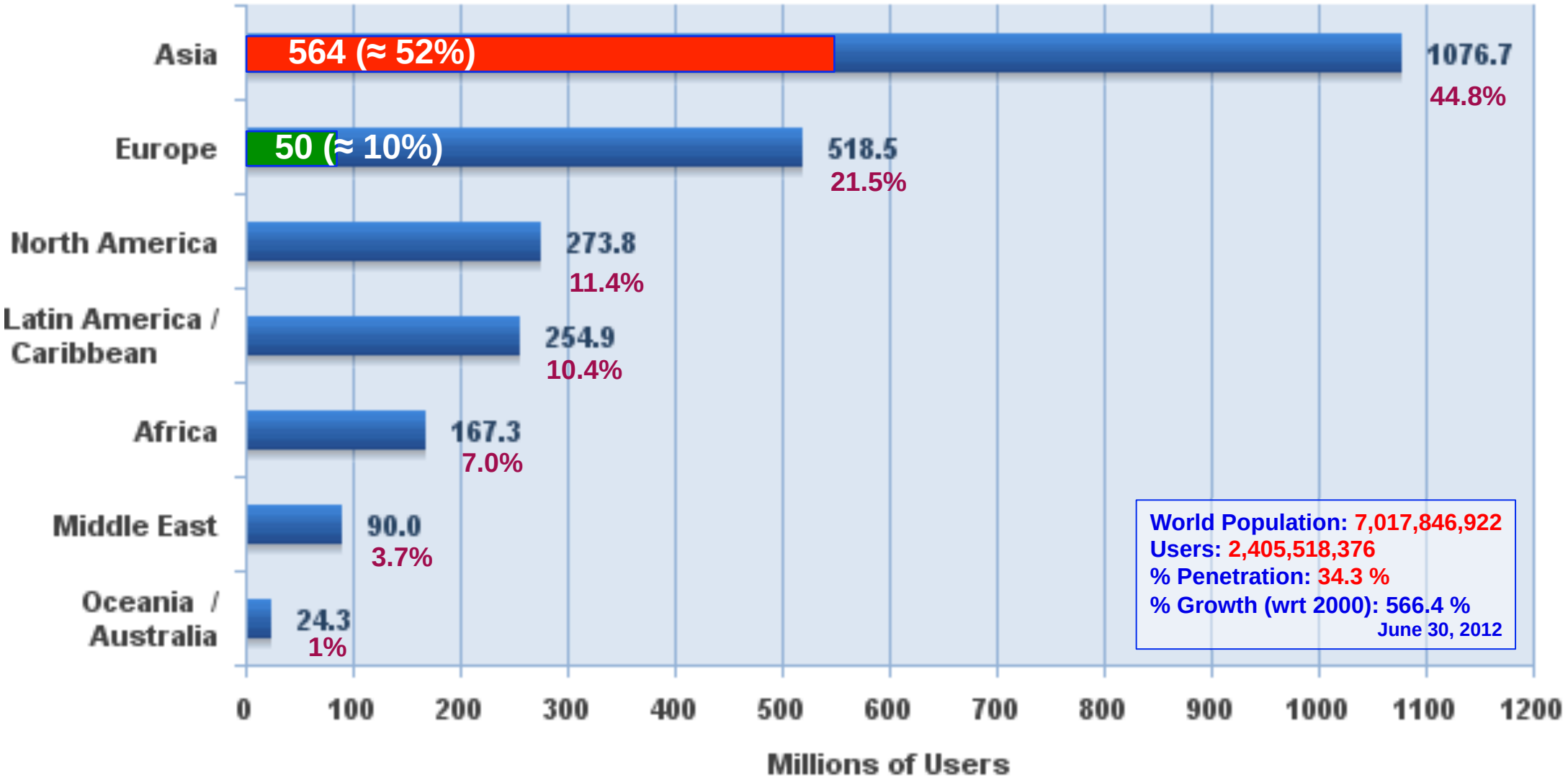
- **Enhanced Functionalities and Complexity**
- **Economic Pressure —> reuse (COTS components)**
- **Intrusions, Attacks,…**

Availability		Unavailability per year
6 x '9'	0,999999	32s
5 x '9'	0,99999	5mn 15s
4 x '9'	0,9999	52mn 34s
3 x '9'	0,999	8h 46mn
2 x '9'	0,99	3d 16h
1 x '9'	0,9	36d 12h

From: J. Gray, Dependability in the Internet Era, Stanford, 2006

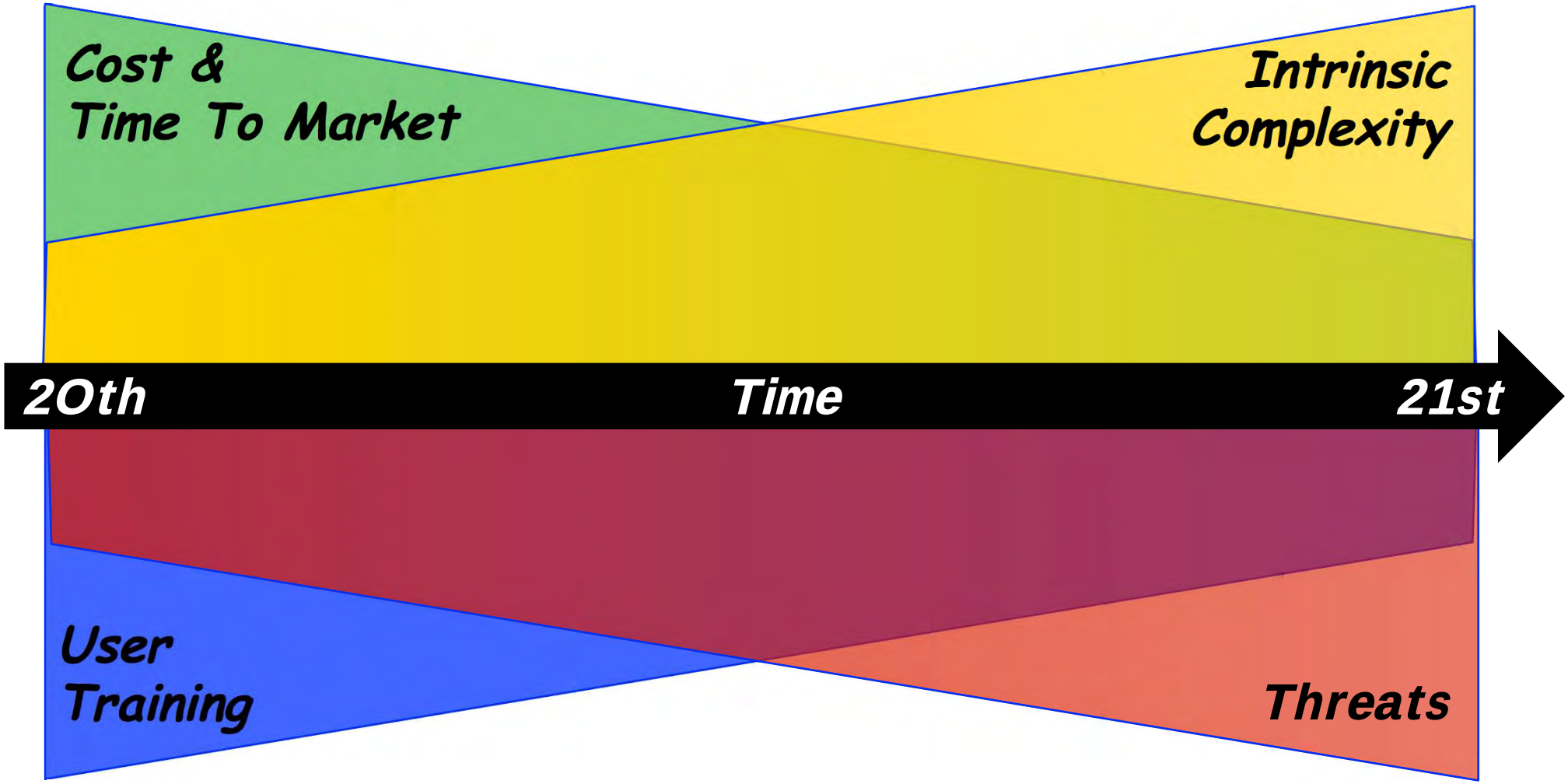
Internet Usage — Worldwide

China
France



Source: Internet World Stats - www.internetworldstats.com/stats.htm
2,405,518,376 Internet users estimated for June 30, 2012

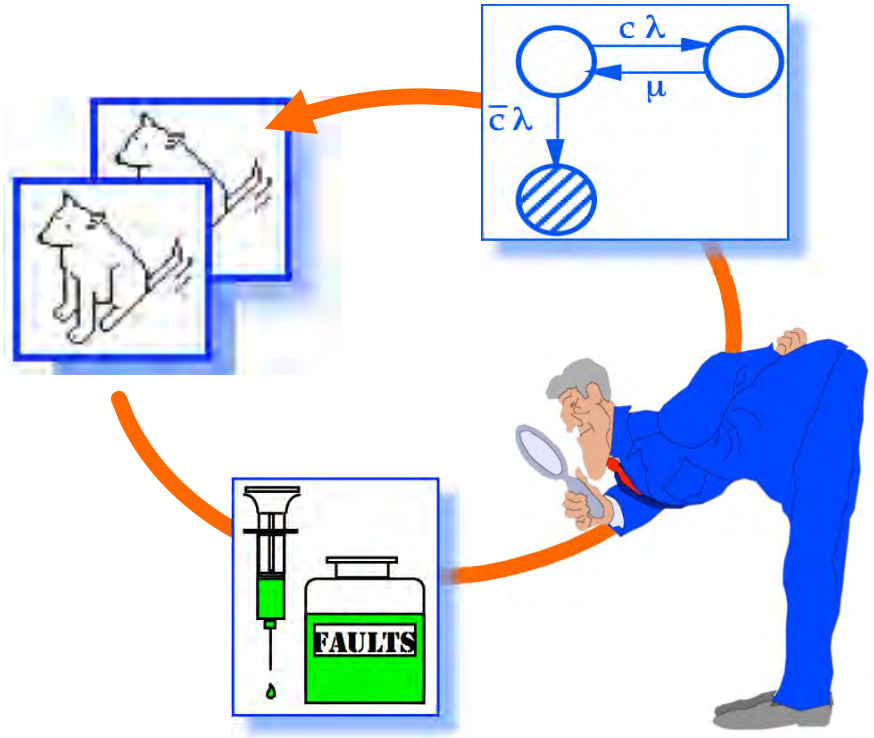
Looking Ahead: An Ever Moving Target



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

=> Dependable Computing

- Terminology and Basic Concepts
- Architecting Dependable Systems: Fault Tolerance
- Dependability Assessement : Modeling, Testing, Benchmarking
- Conclusions and Perspectives



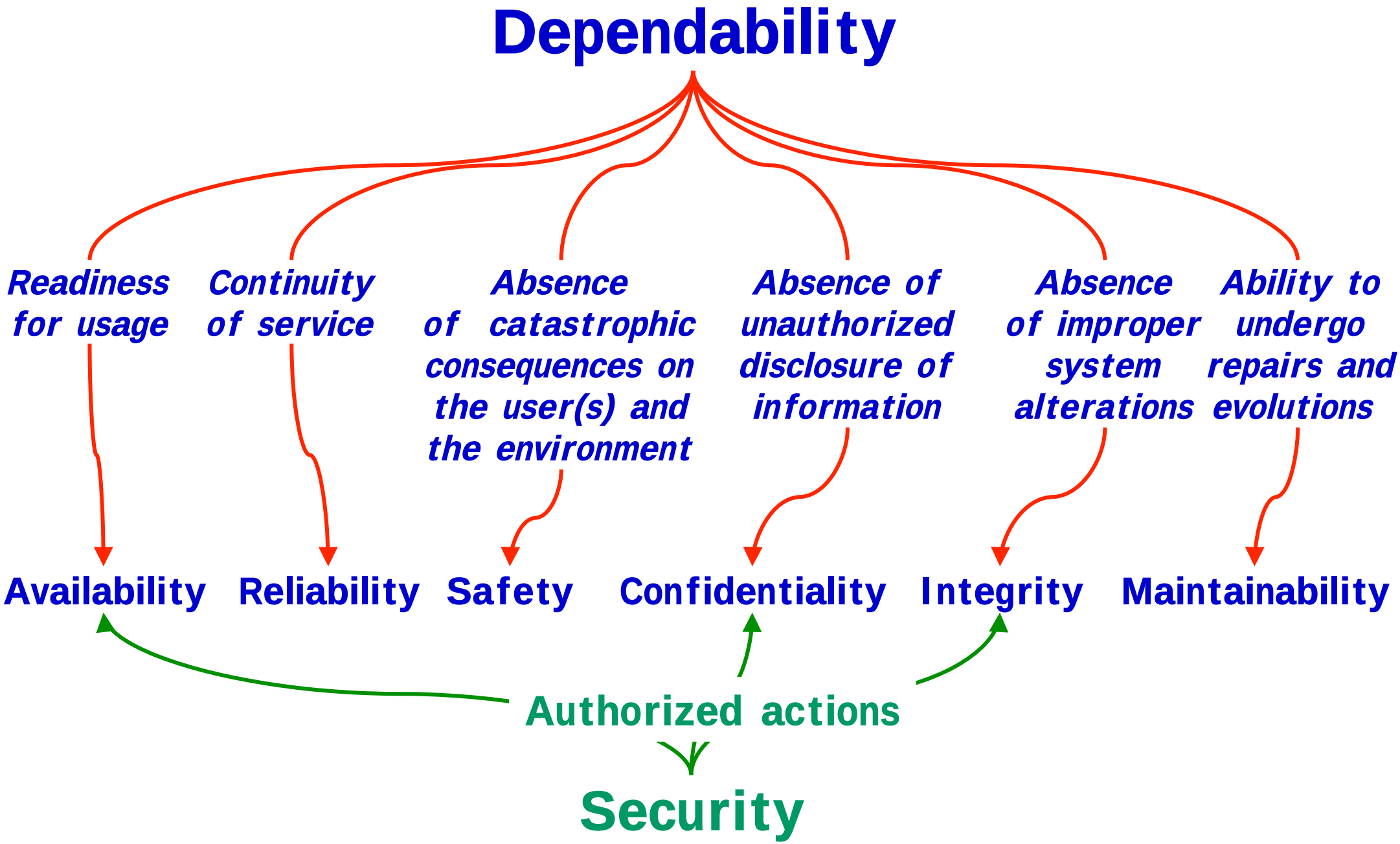
About Dependability

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)

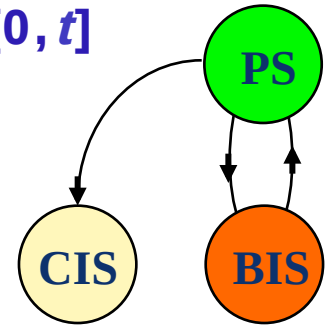
- When failures are more frequent or more severe than acceptable: **dependability failure**


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 $[0, t]$

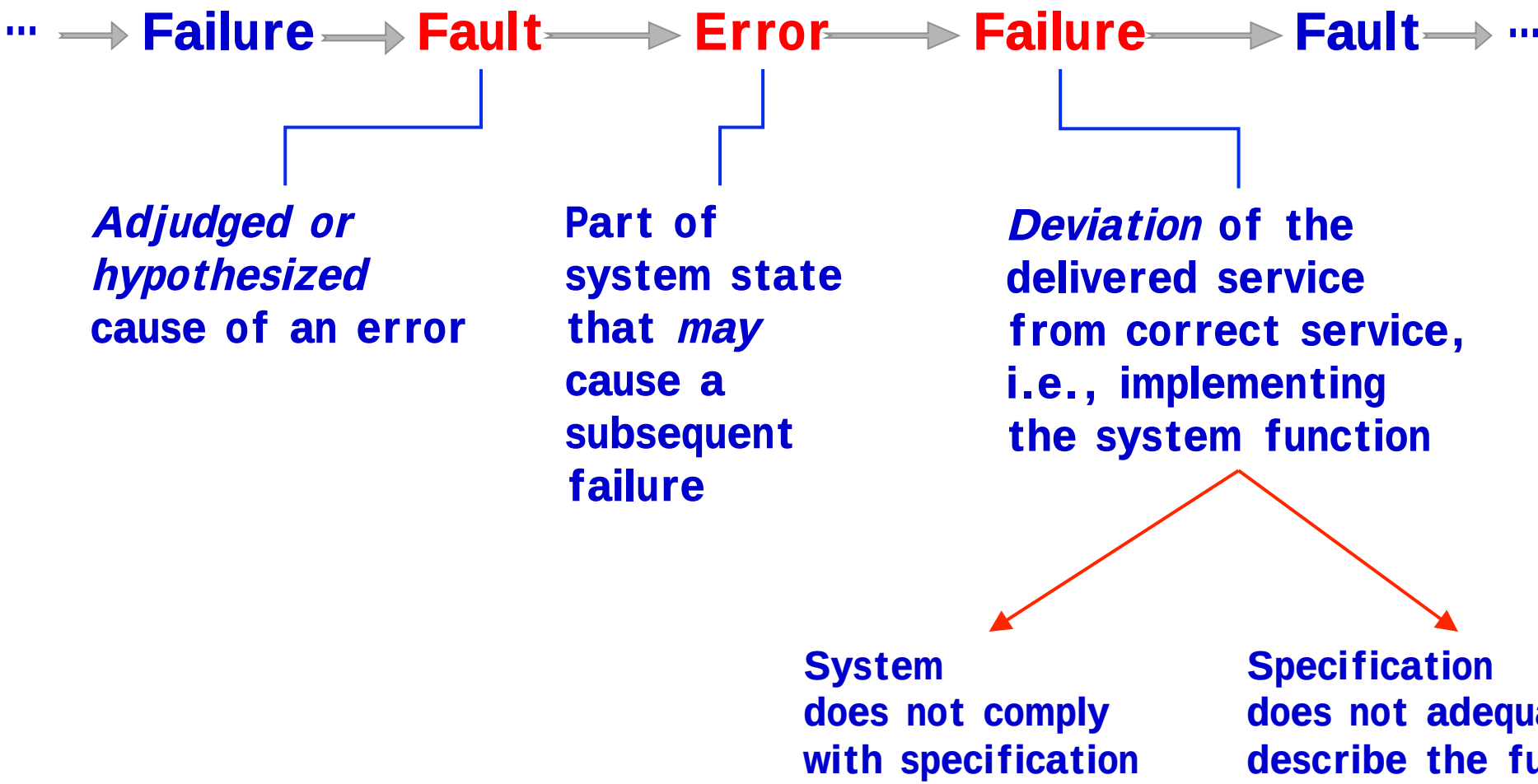
  [Analogous to reliability, but concerned with catastrophic failures]



- ***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

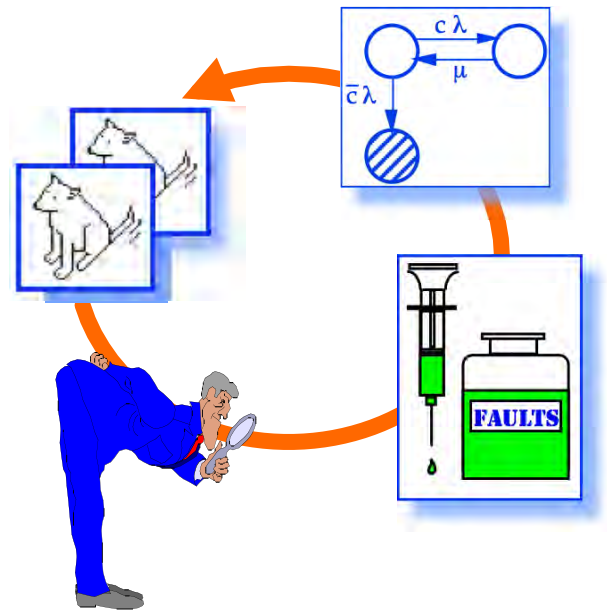


The "Dependability Tree" *

Dependability

- *Attributes*
  - Availability
  - Reliability
  - Safety
  - Confidentiality
  - Integrity
  - Maintainability

  (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

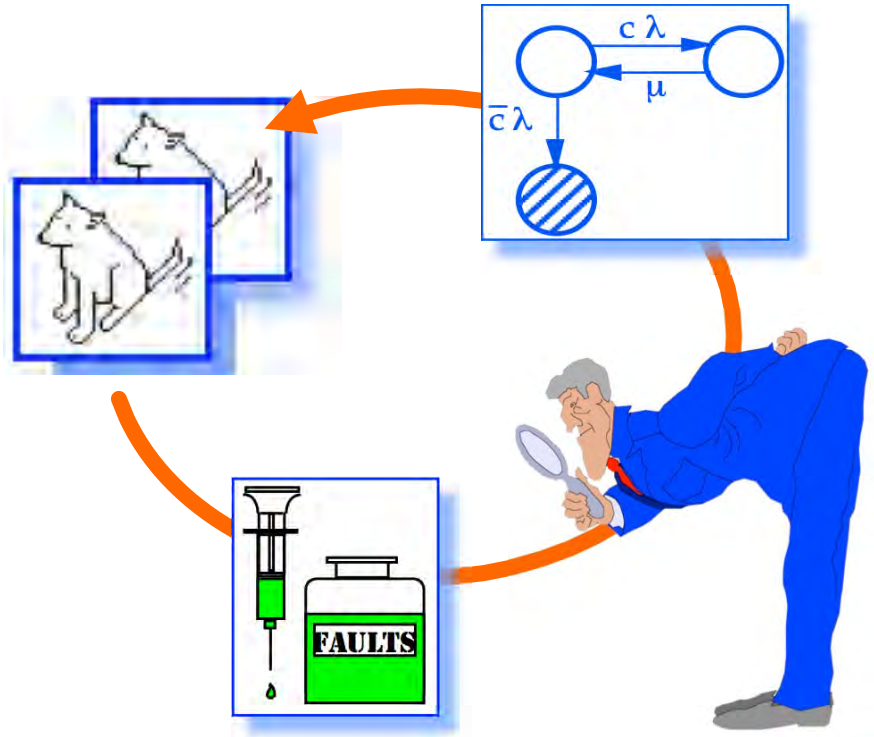
When delivered service deviates (value, timing) from implemented function

↓

Failure

=> Dependable Computing

- Terminology and Basic Concepts
- **Architecting Dependable Systems: Fault Tolerance**
- Dependability Assessement : Modeling, Testing, Benchmarking
- Conclusions and Perspectives



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

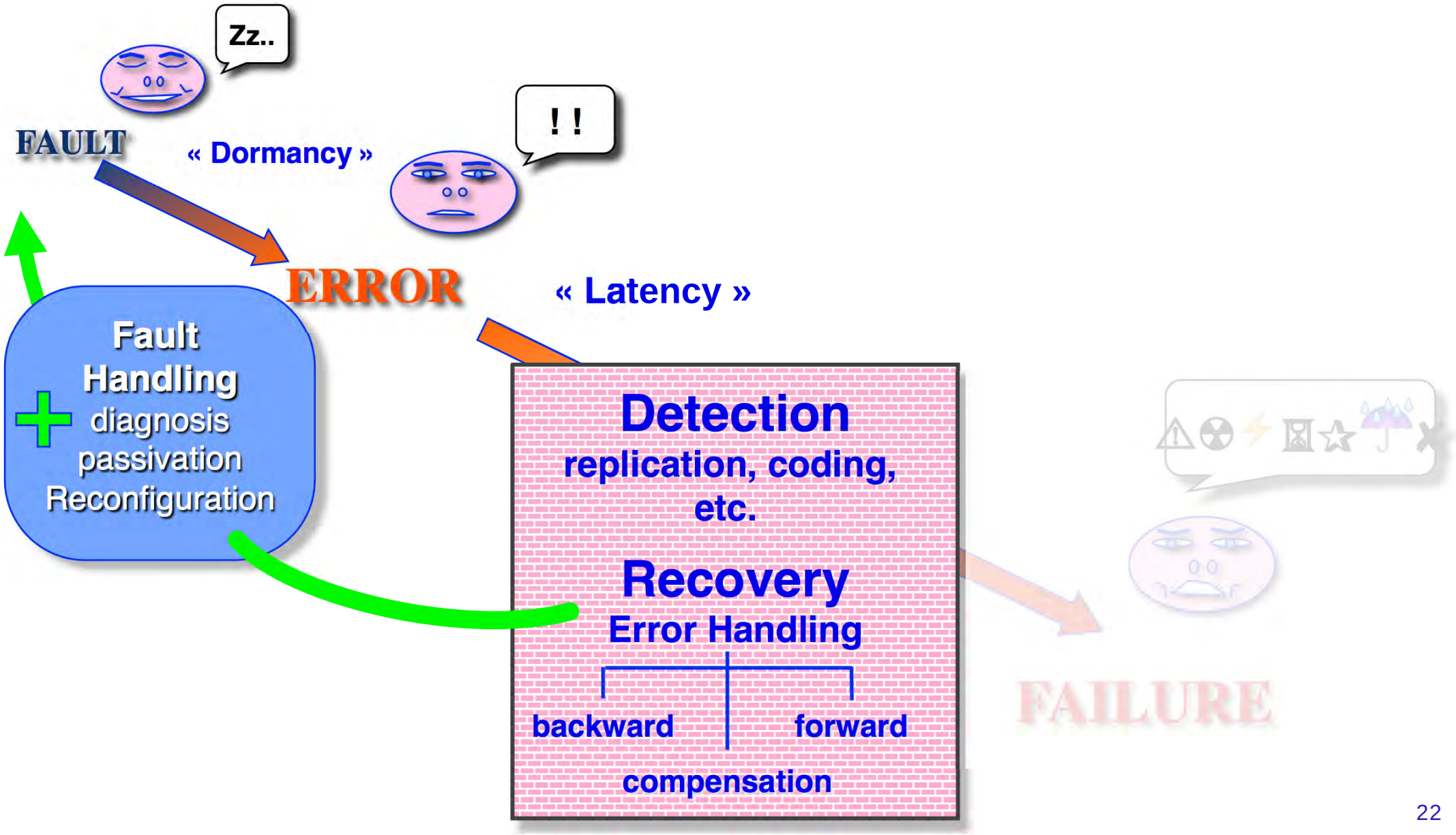
- **Diagnosis**: identifies and records the error cause(s), according to localisation and category
- **Isolation**: performs physical or logical exclusion of the fauty component(s) from further contribution to service delivery, i.e., makes the fault(s) dormant
- **Reconfiguration**: either switches in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

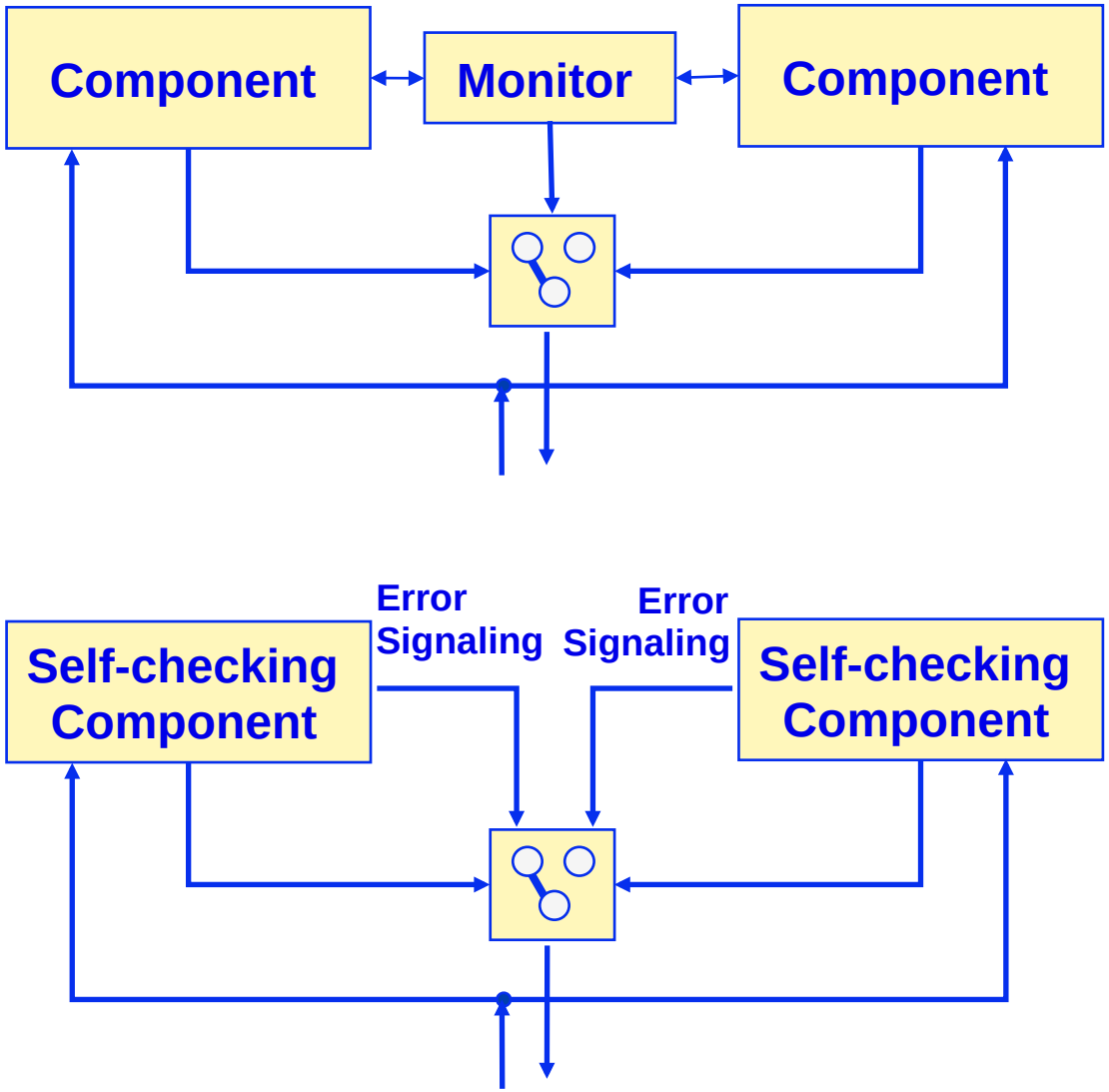


Impact of Fault Tolerance

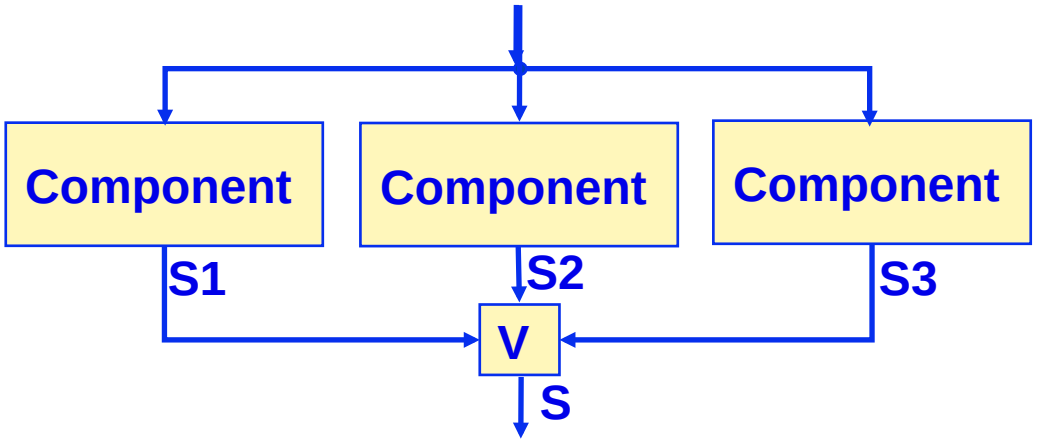
Dependability ≈ 1 - Pr{fault} × Pr{error/fault} × Pr{failure/error}

↓ 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 Redundancy (Active Duplex)



Static Redundancy: Triple Modular Redundancy

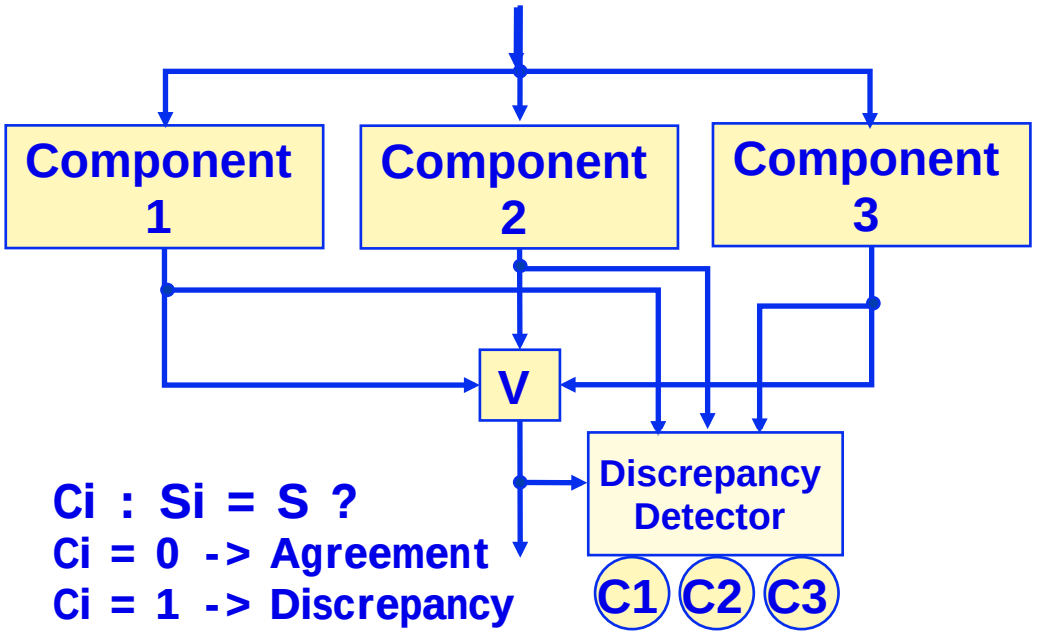


S = MAJ (S1,S2,S3)

- 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

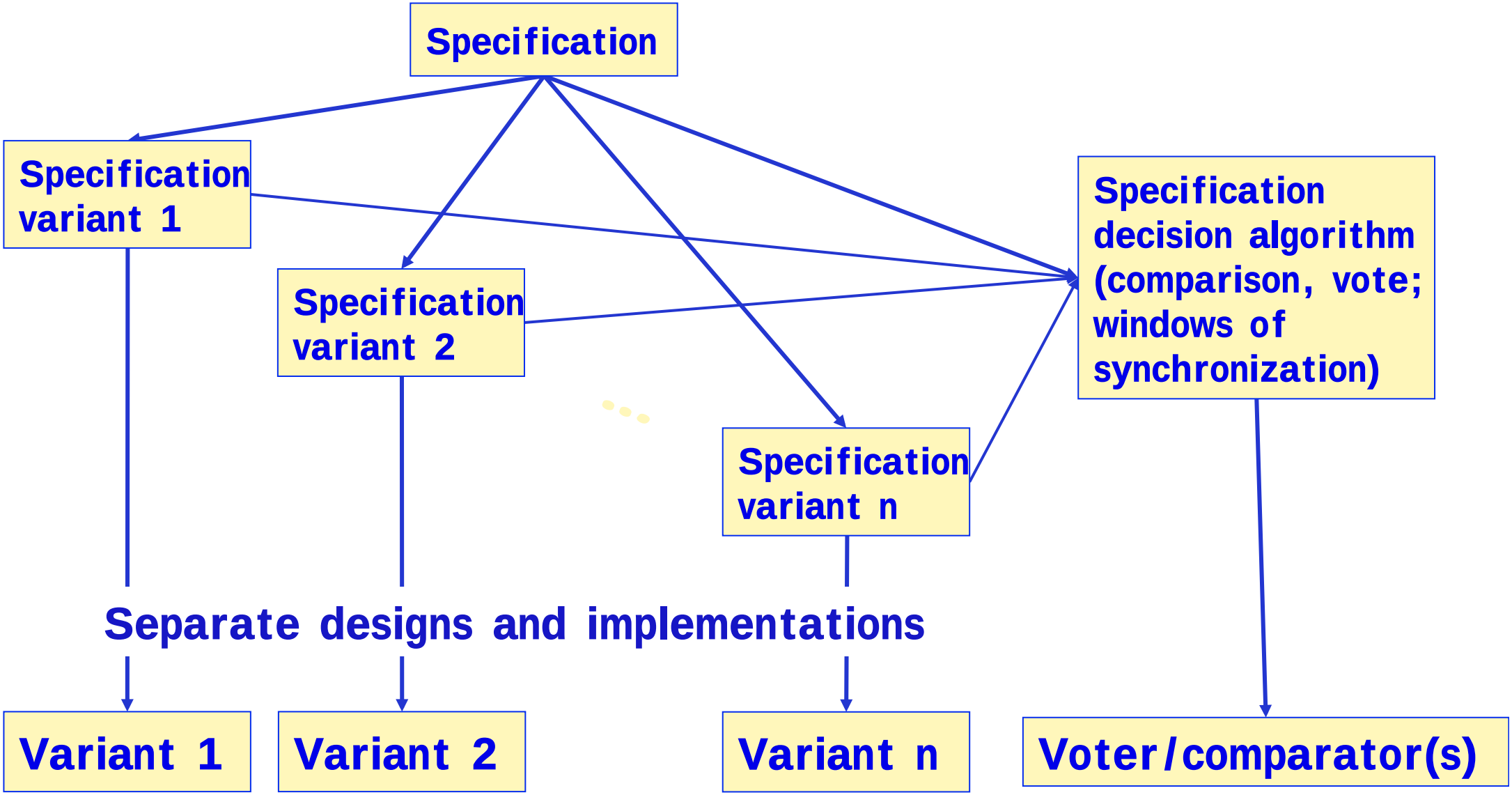
$S=(S1 \cap S2) \cup (S2 \cap S3 \cup (S1 \cap S3)$



	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



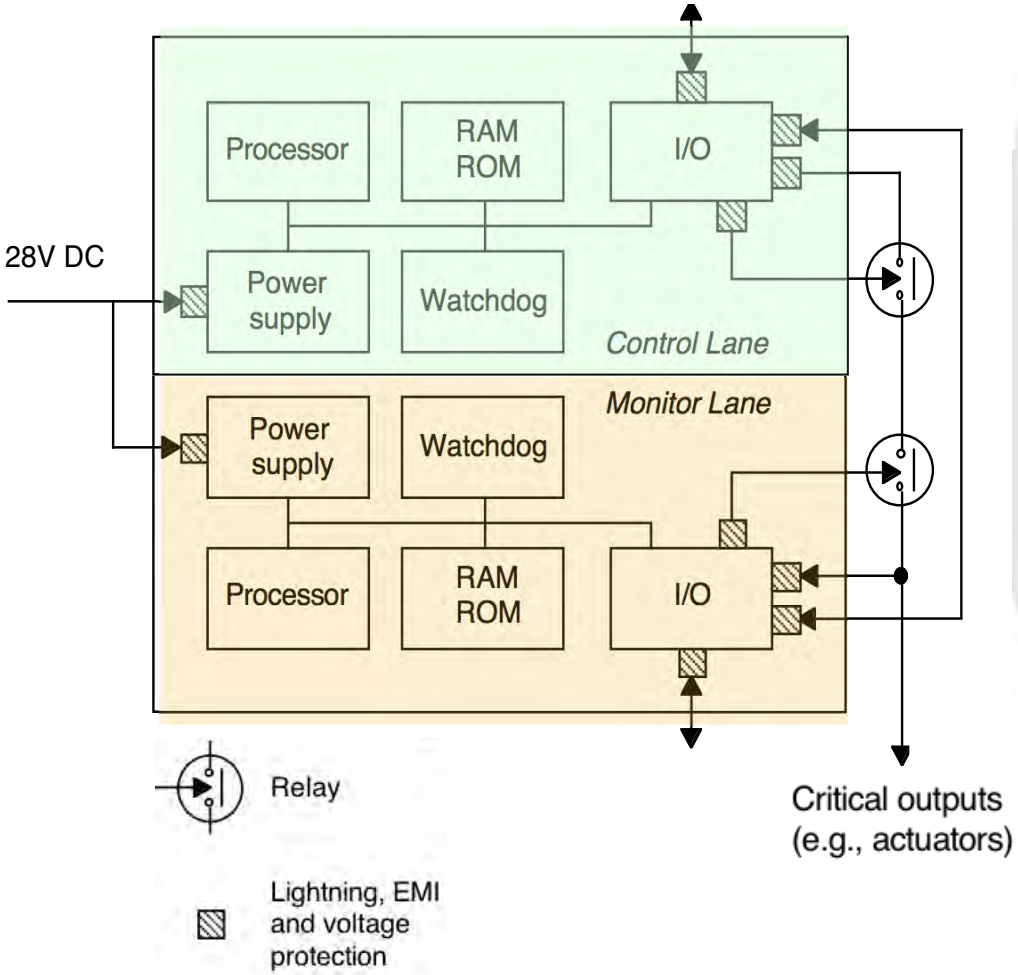
Design Diversity

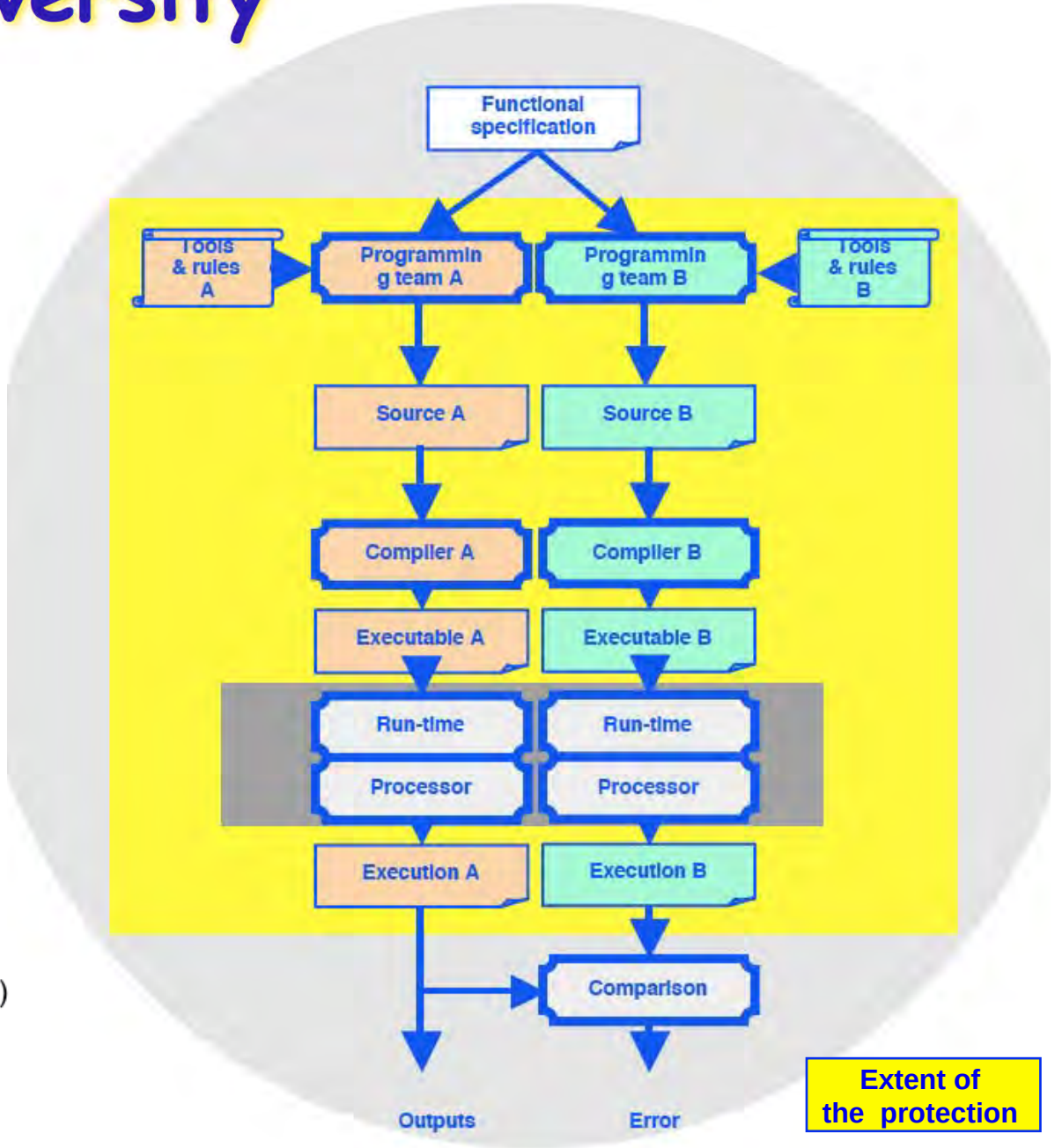
- **Aim: fault independency (↘ risk of common mode failures)**
  Issues: common specification, inter-variant synchronization & decision
- **Major techniques:**
  - Recovery Blocks
  - N-Version Programming
  - N-Self-Checking Prgramming
- **Operational use**
  - **Civil aviation: generalized, at differing 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)





Aircraft Maintenance: Current Scenario

Pilot

Onboard equipment

Maintenance engineer

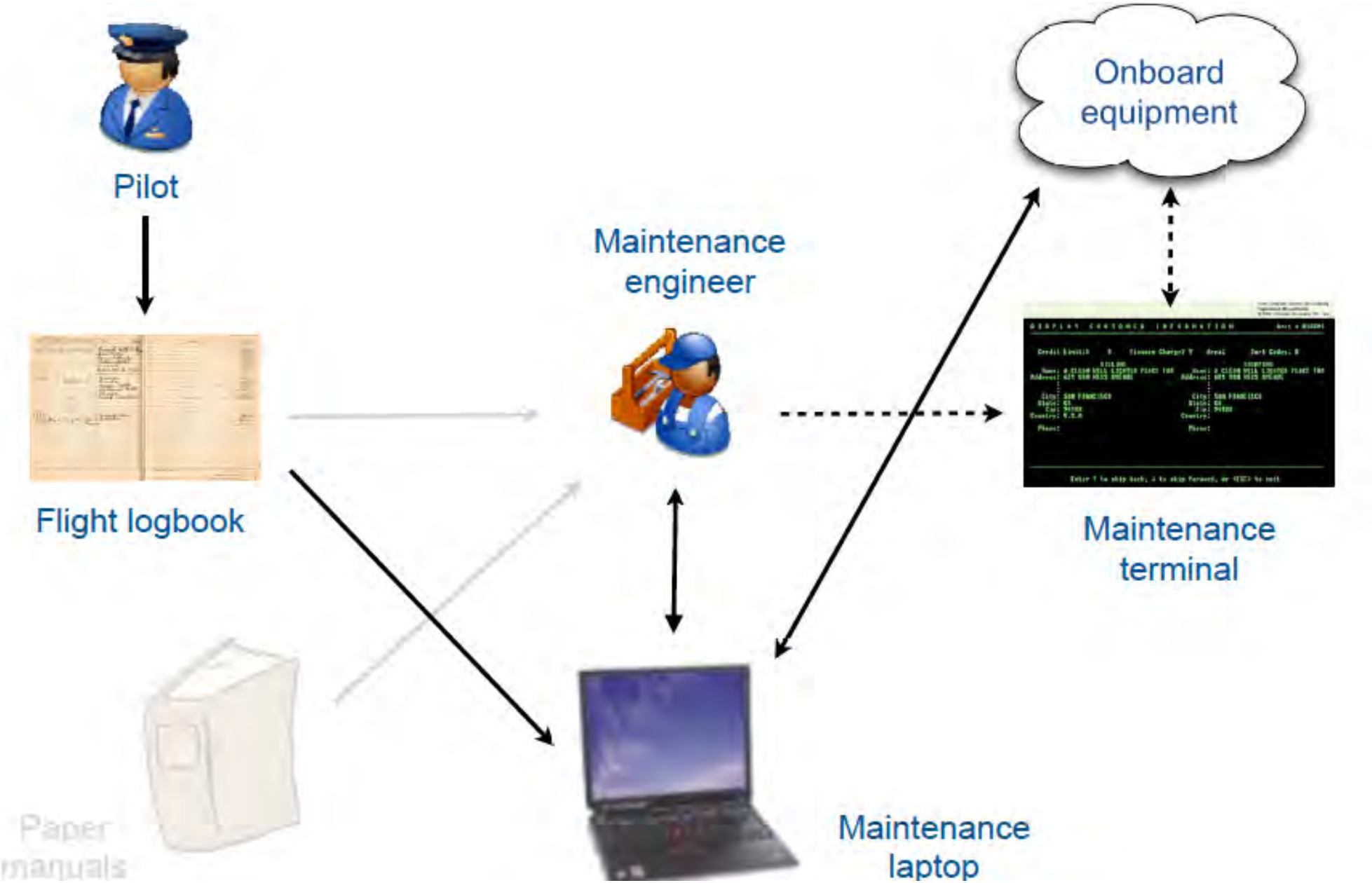
Flight logbook

Maintenance terminal

Paper manuals

Electronic manuals

Aircraft Maintenance: Laptop Scenario



Connecting a Laptop?

Connecting a Laptop?

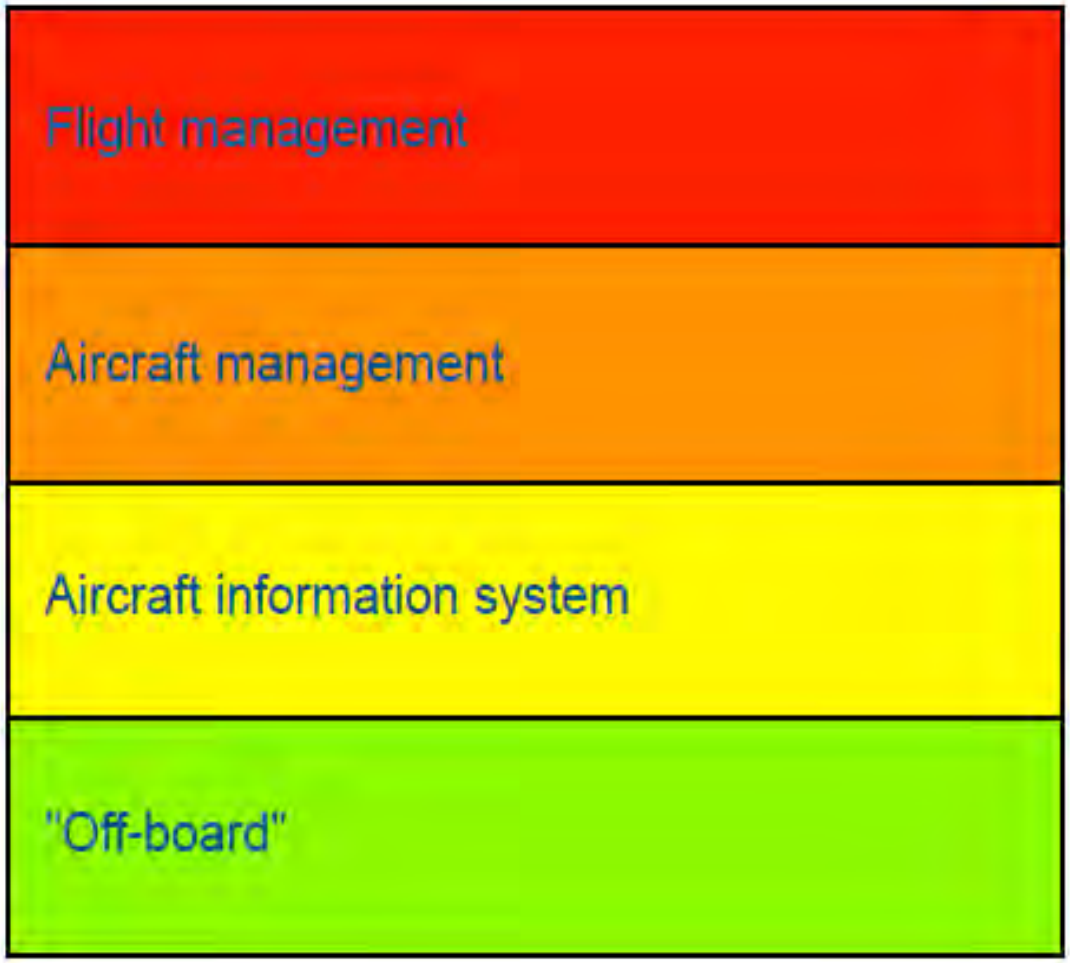
Execution confidence

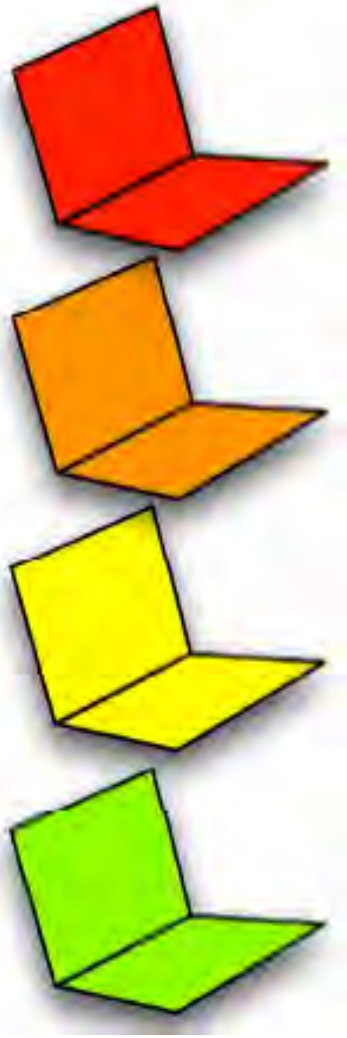
++

++

+

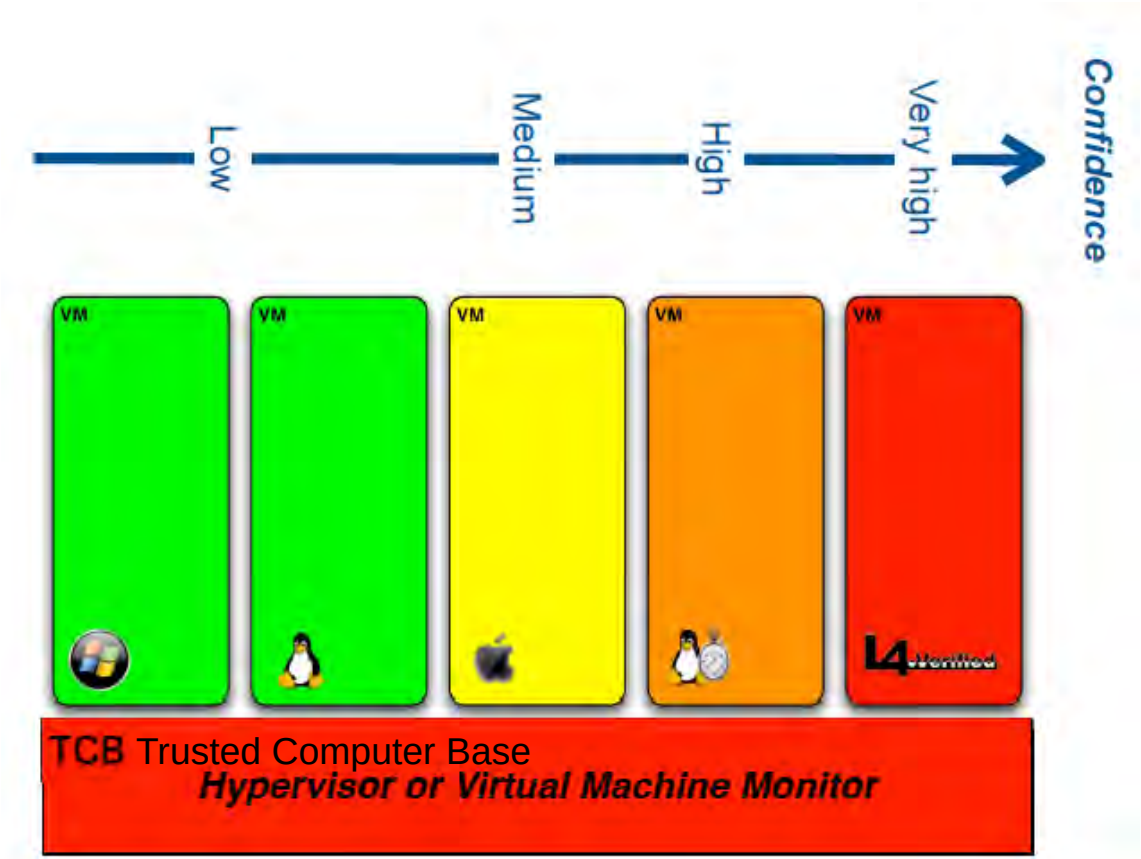
-



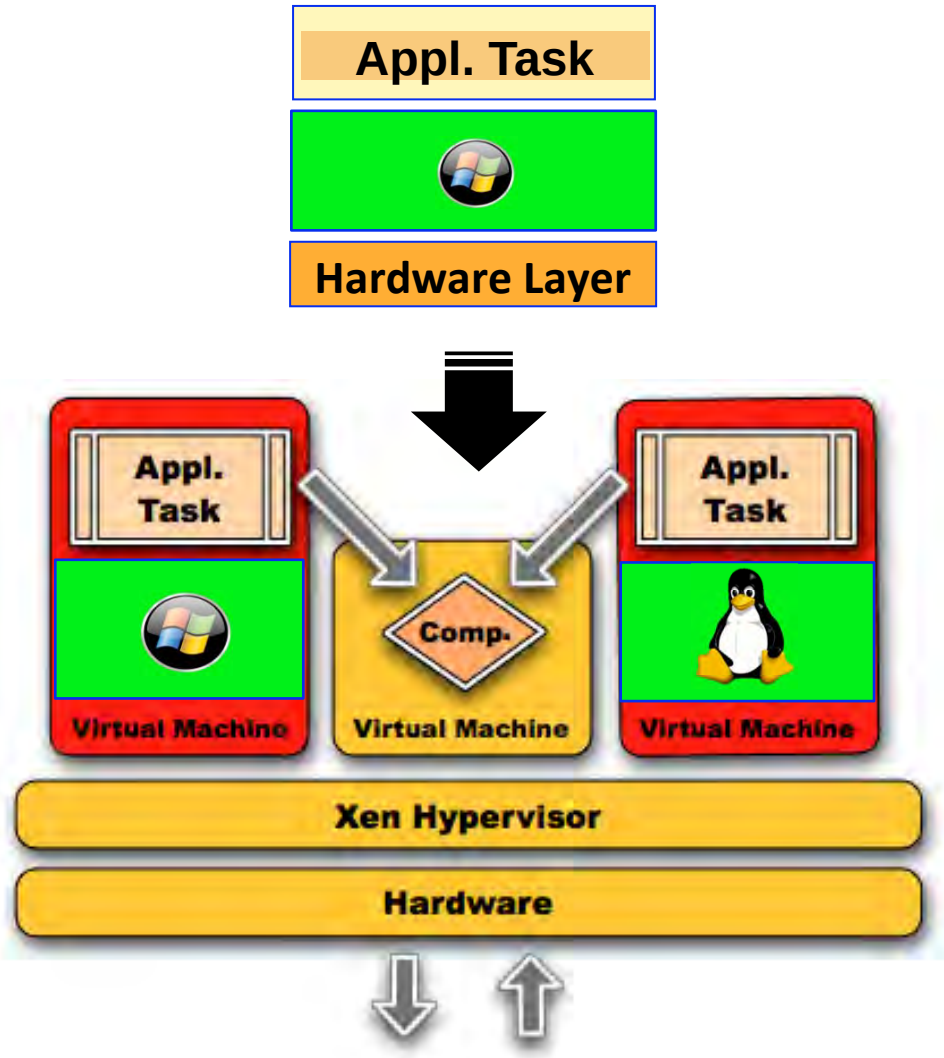


Virtualization for Dependability

Partitioning and Segregation



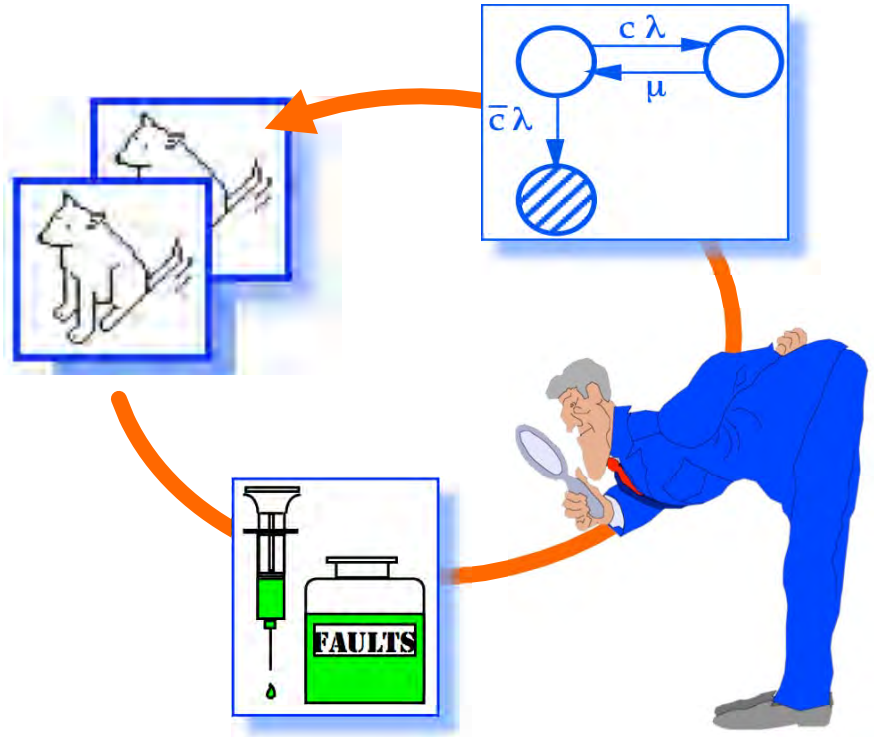
Diversified Duplex



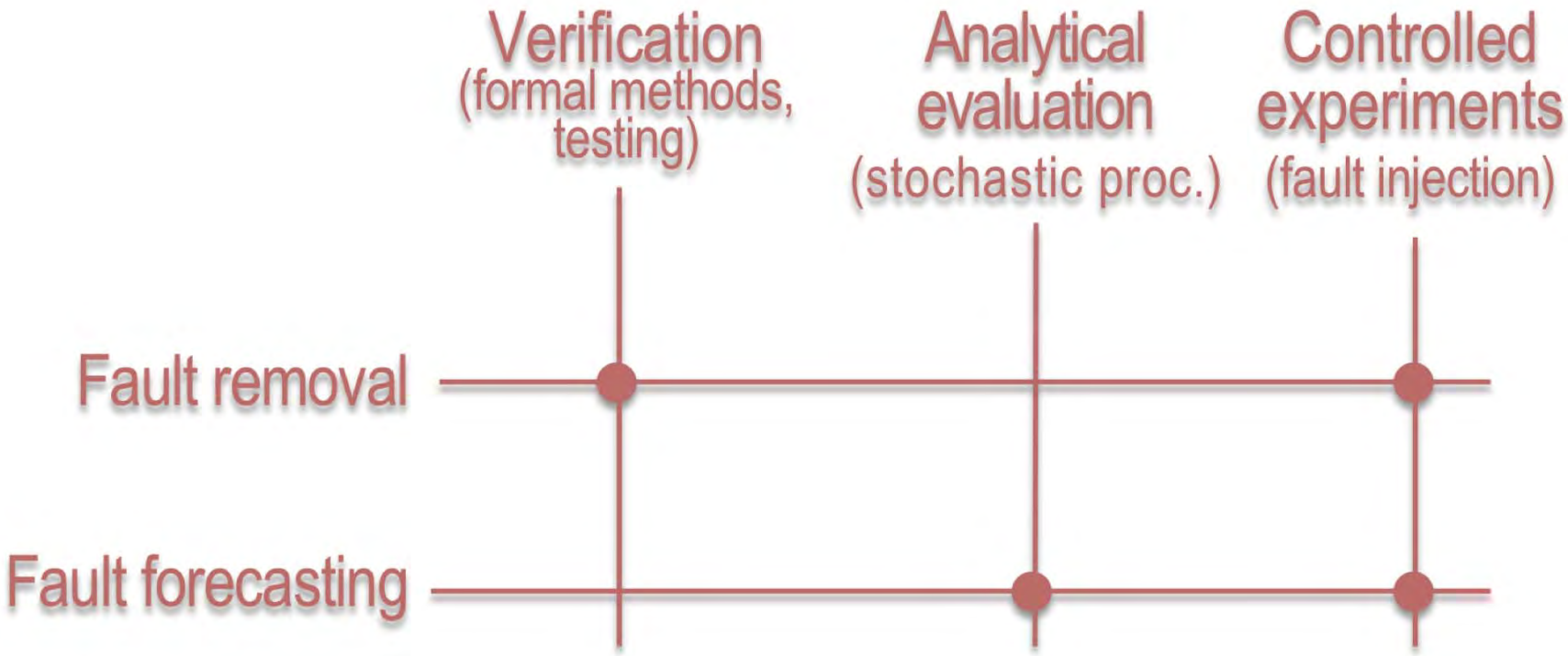
Y Laarouchi, Y Deswarte, D Powell, J Arlat, E de Nadai
Connecting Commercial Computers to Avionics Systems
28th IEEE/AIAA Digital Avionics Systems Conference, DASC'09, 2009

=> Dependable Computing

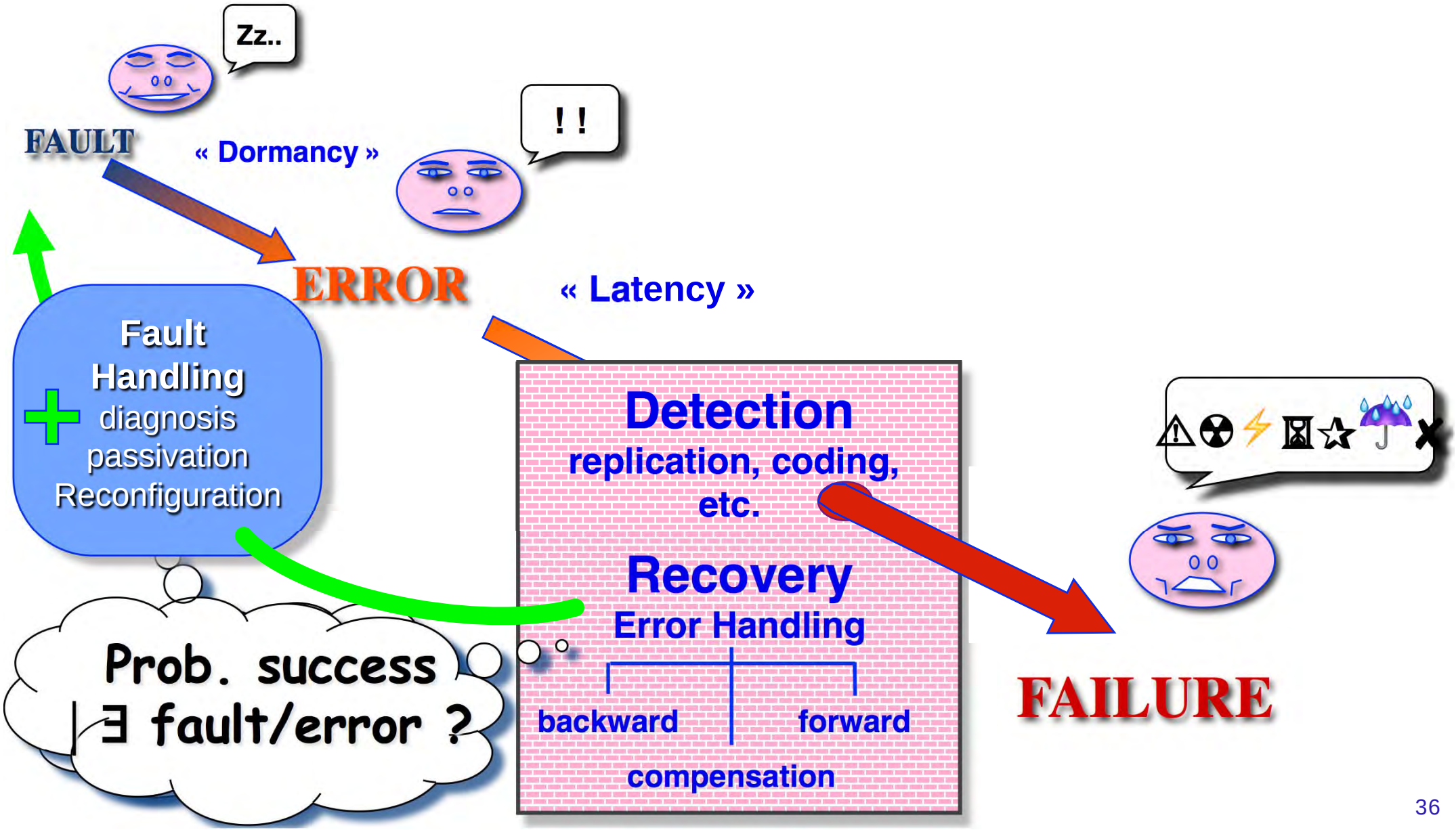
- Terminology and Basic Concepts
- Architecting Dependable Systems: Fault Tolerance
- **Dependability Assessement : Modeling, Testing, Benchmarking**
- Conclusions and Perspectives



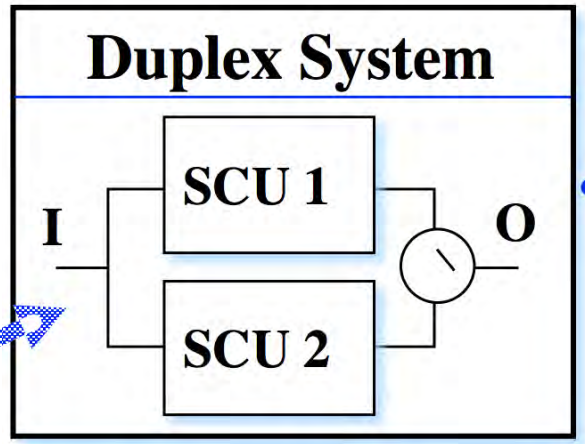
Despendability Assesments Methods

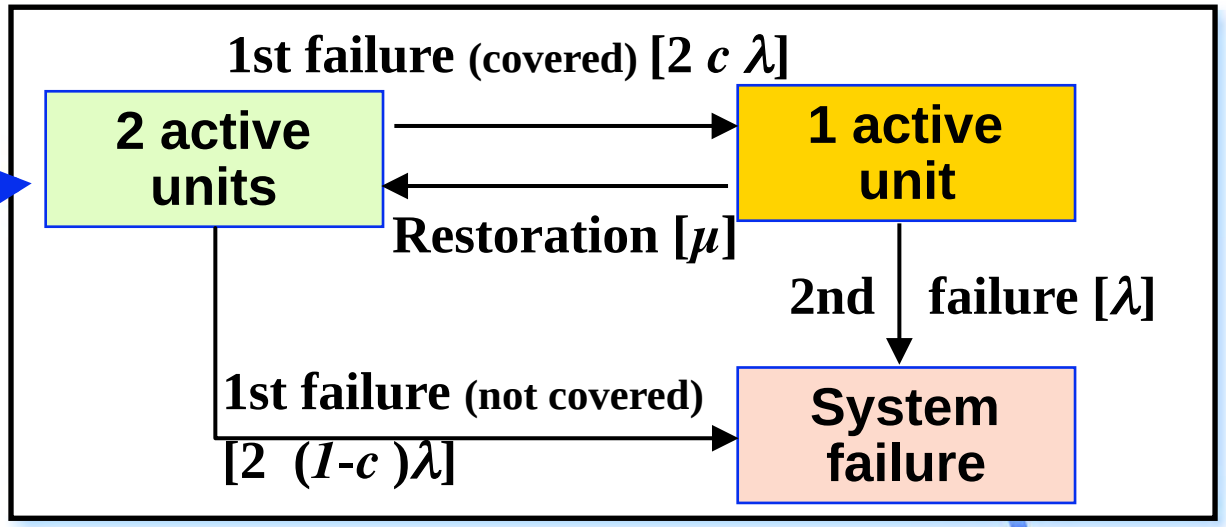


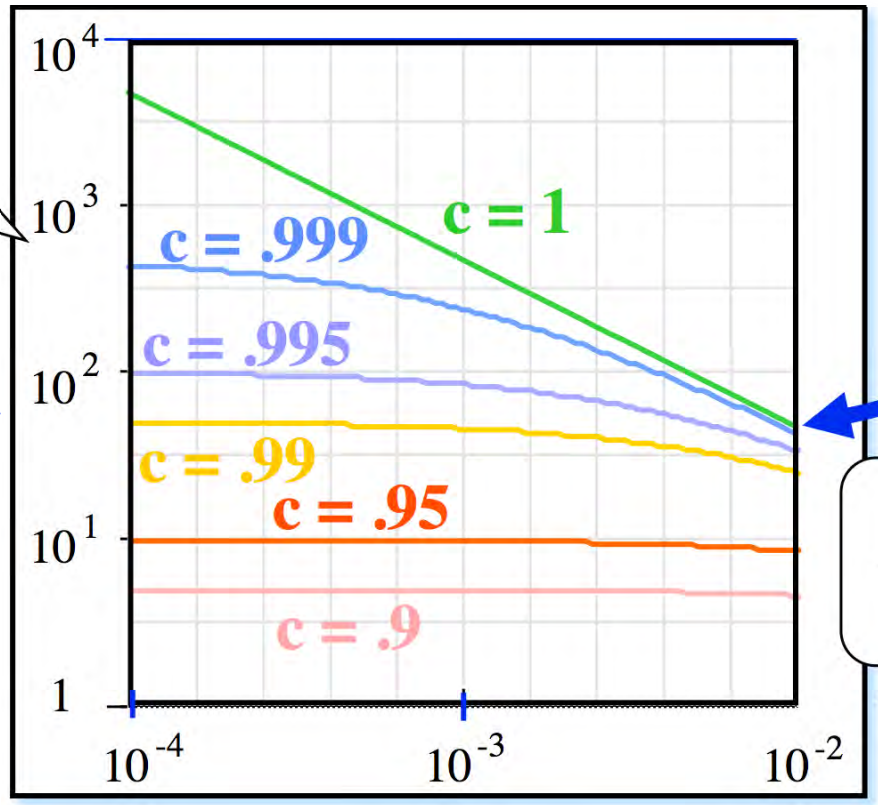
Fault Tolerance … and Coverage



Impact of Coverage on Dependability





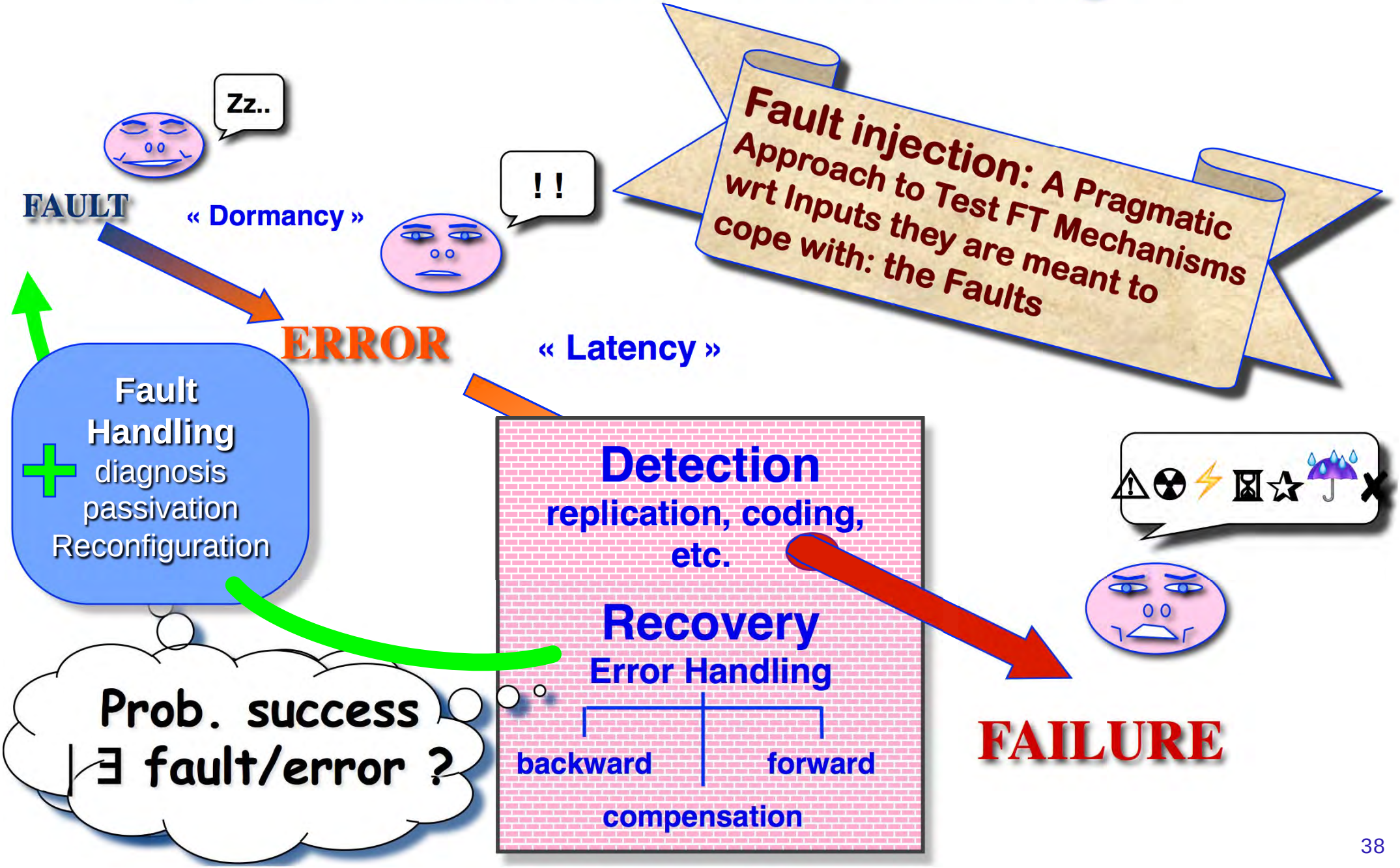


$$\frac{MTTF_{Syst.}}{MTTF_{Unit.}}$$

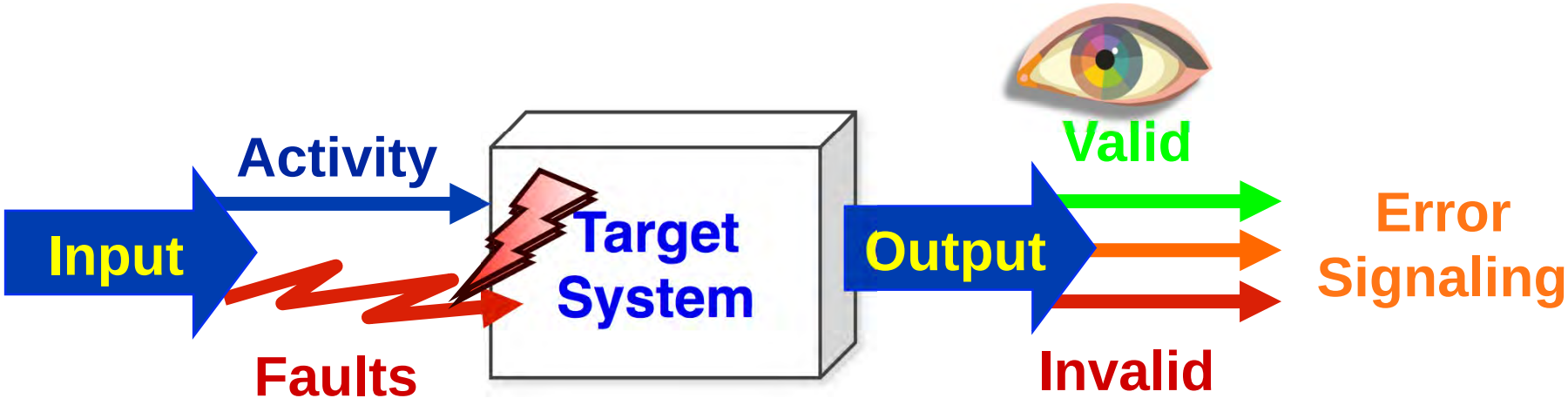
$$\frac{MTTR_{Comp.}}{MTTF_{Comp.}}\left(\frac{\lambda}{\mu}\right)$$

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



Fault Injection-based Assessment

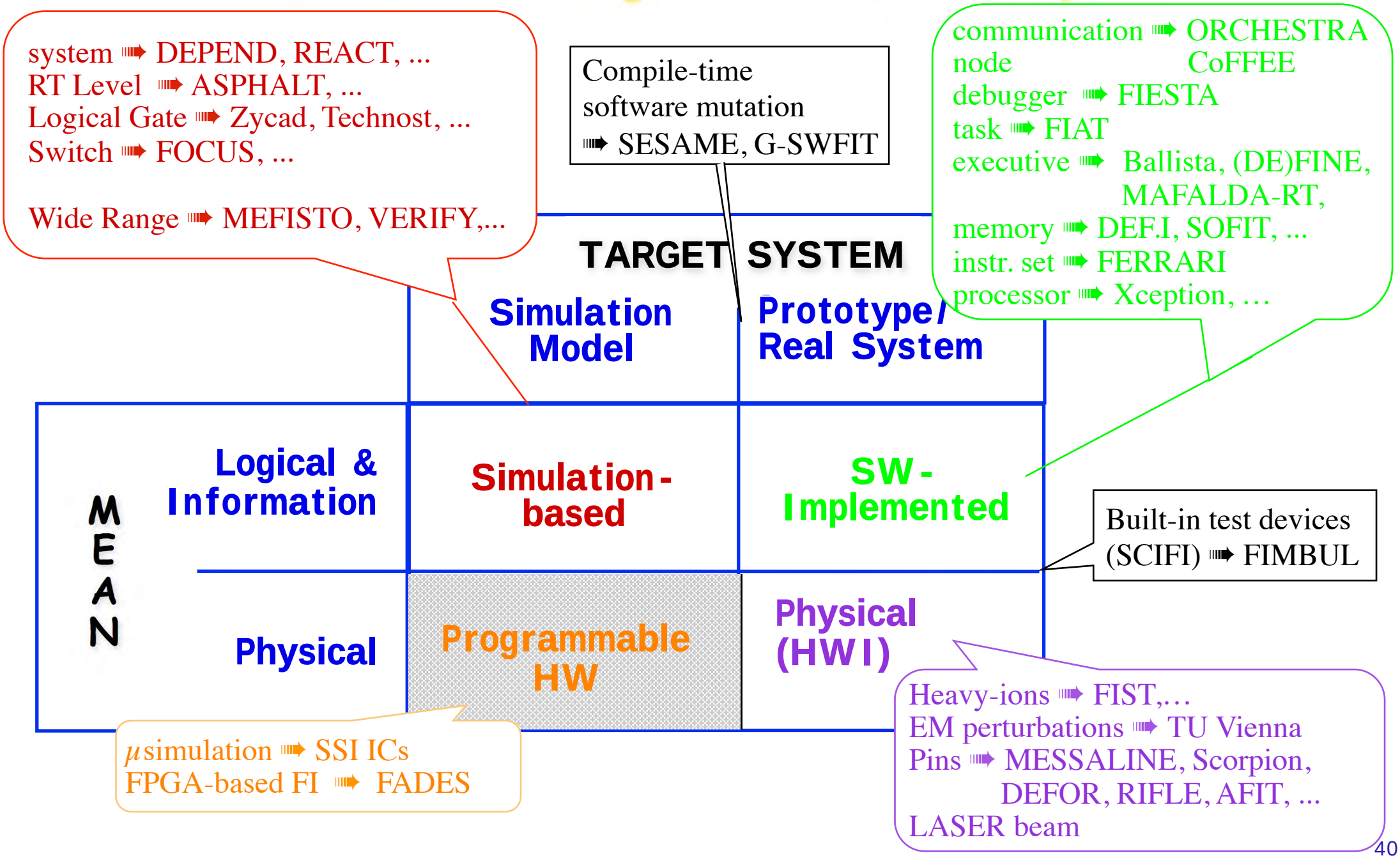


—> Partial dependability assessment:
controlled application of fault/error conditions

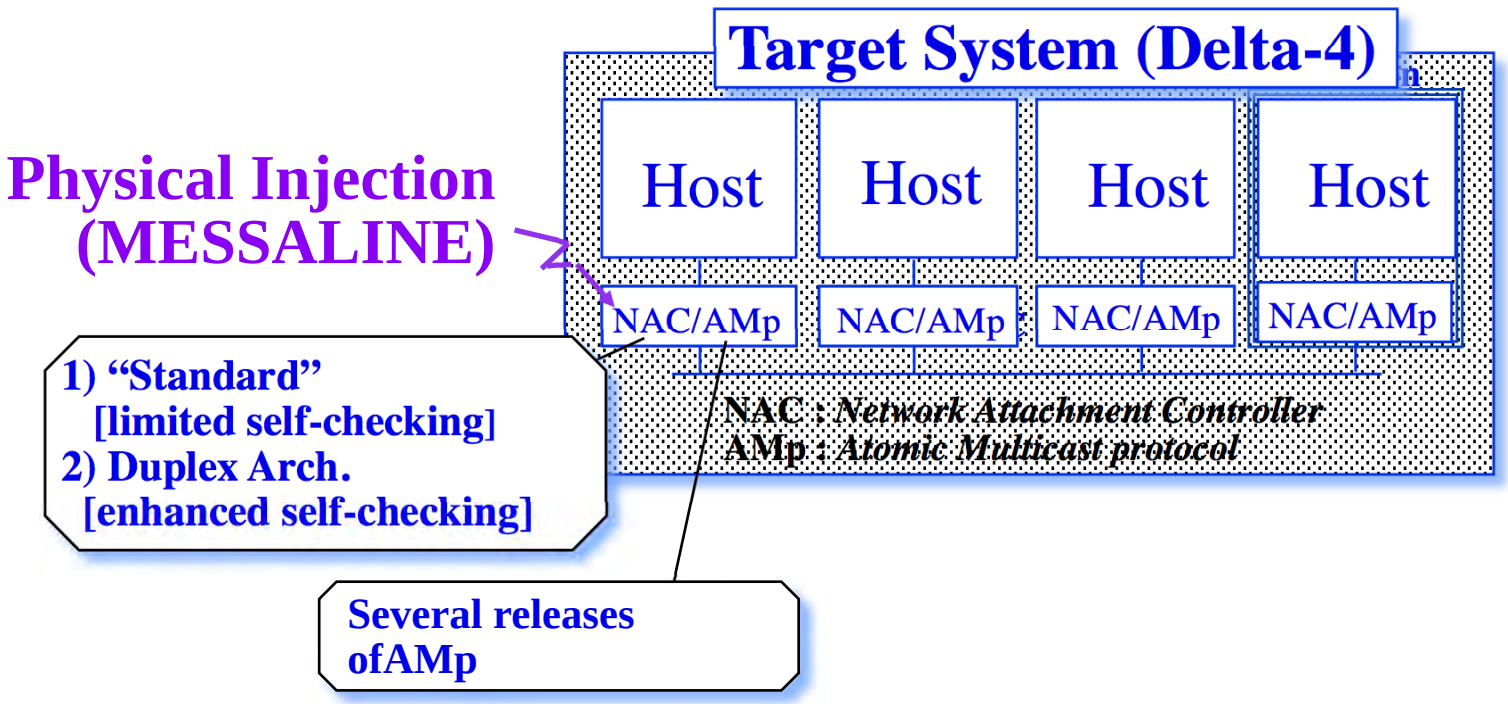
- Testing and evaluation of a fault-tolerant computer system and of its FT algorithms & mechanisms

- Characterization of faulty behaviors & failure modes of several computer systems & components
  -> Dependability benchmarking (comparison purpose)

The Fault Injection Techniques



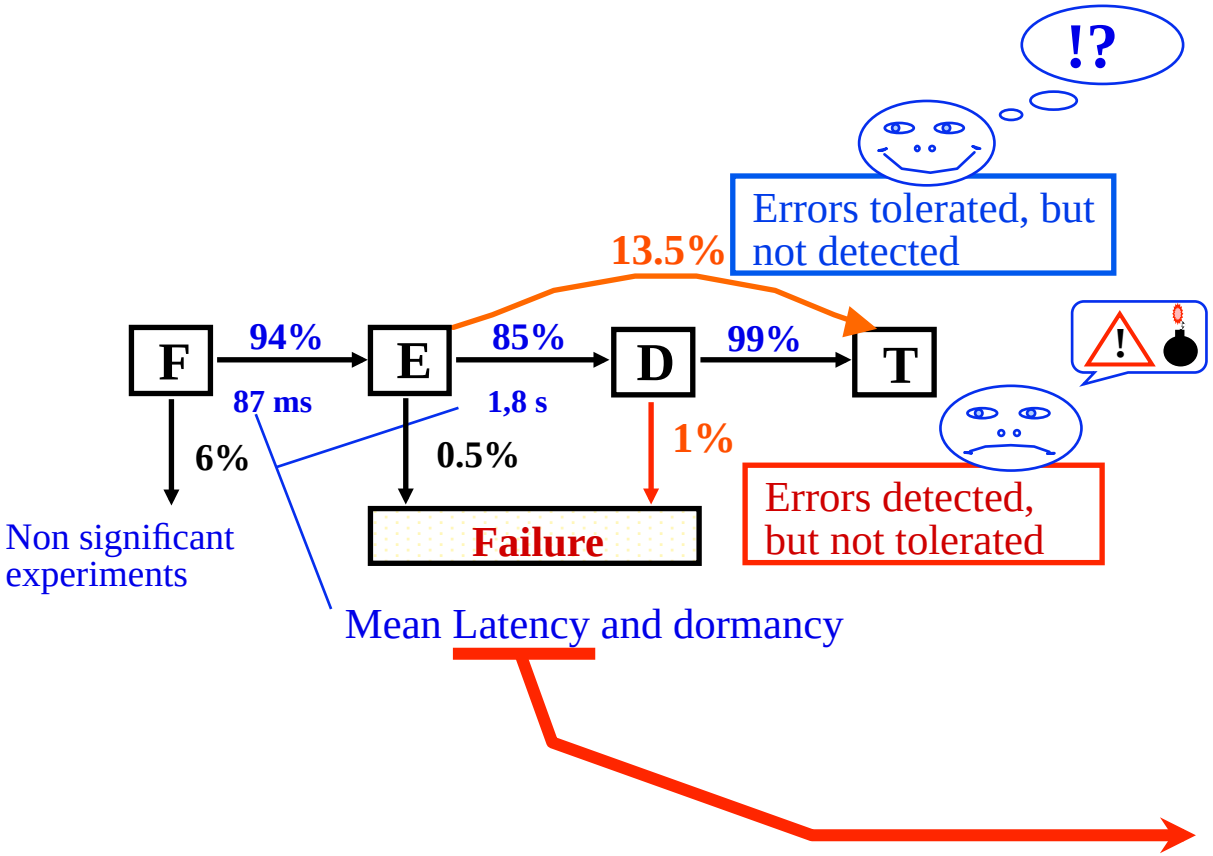
Examples of Experimental Results - 1



T Predicate:
Protocol properties OK and error confined

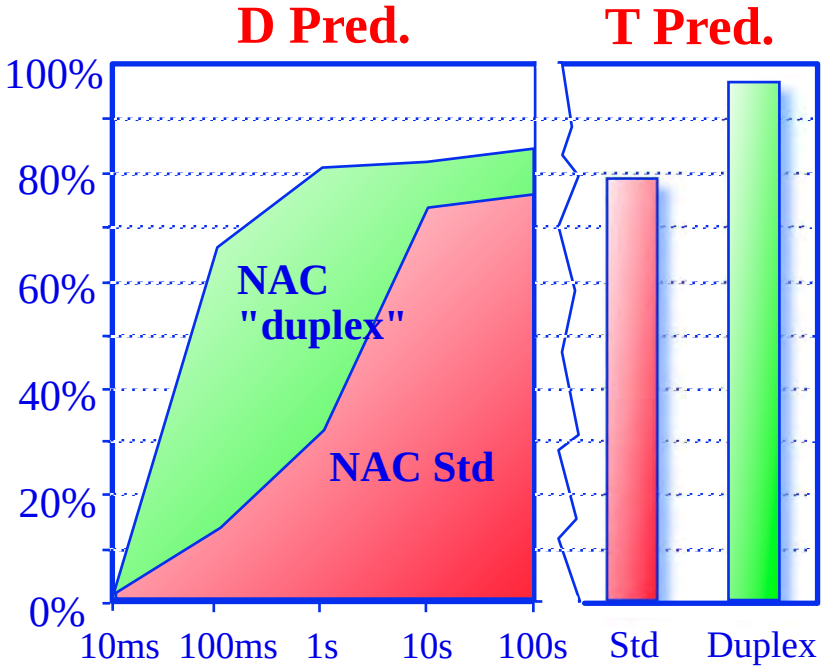
D Predicate:
Self extraction of the injected station

Examples of Experimental Results - 2



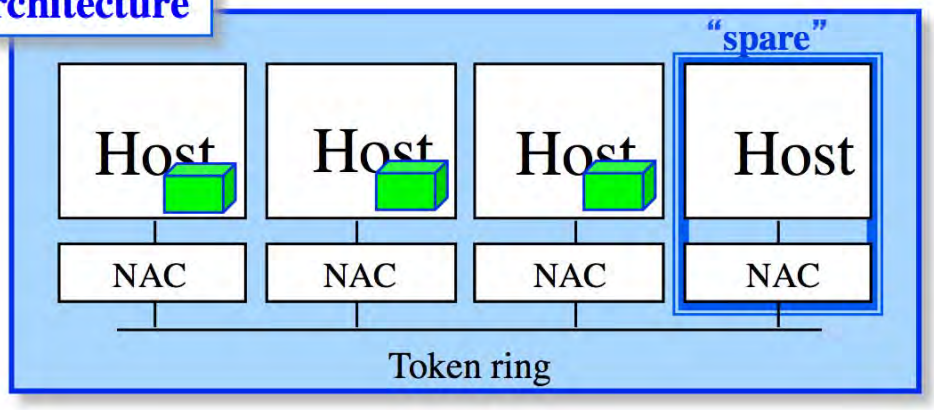
T Predicate:
Protocol properties OK and error confined

D Predicate:
Self extraction of the injected station



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

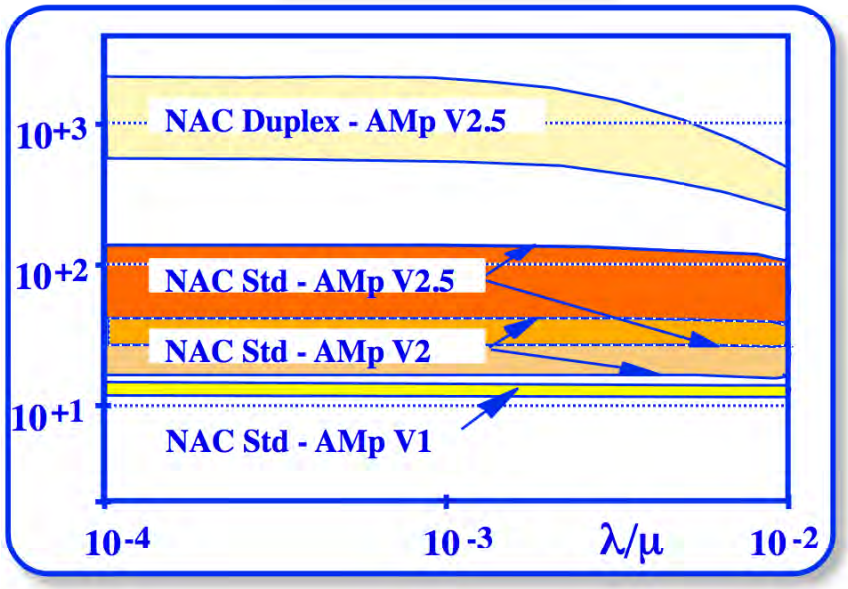
Architecture



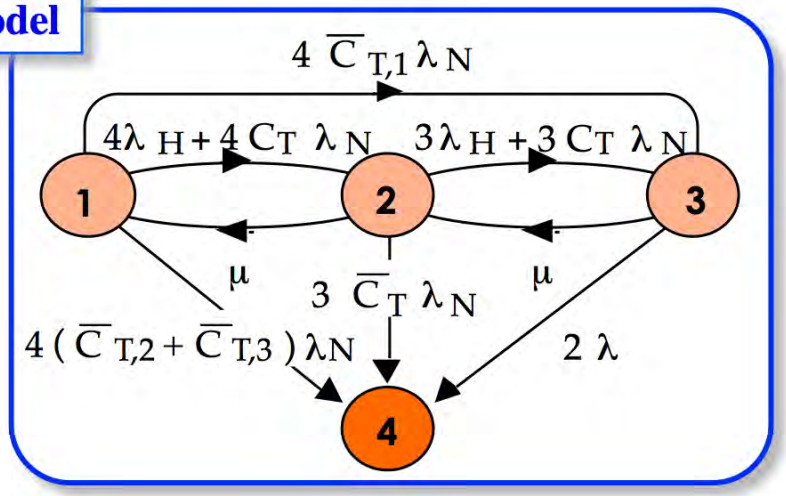
Coverage Factors

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]	45%
[illegible] -AMp V [illegible]	[illegible]	7,79%	1,[illegible]	
NAC Duplex - AMp V 2.5	99,55%	0,32%	0,00%	0,12%

MTFF network
MTFF station

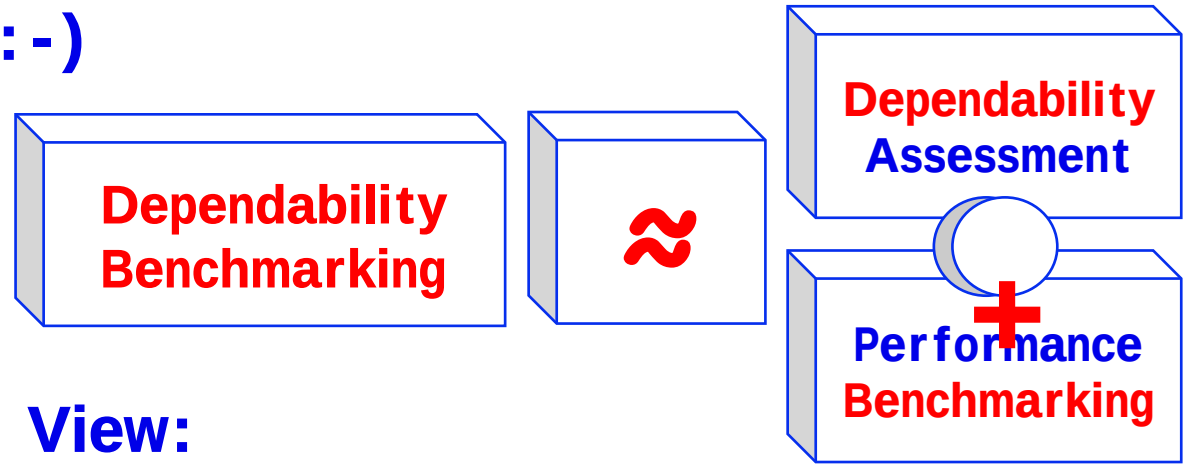


Model

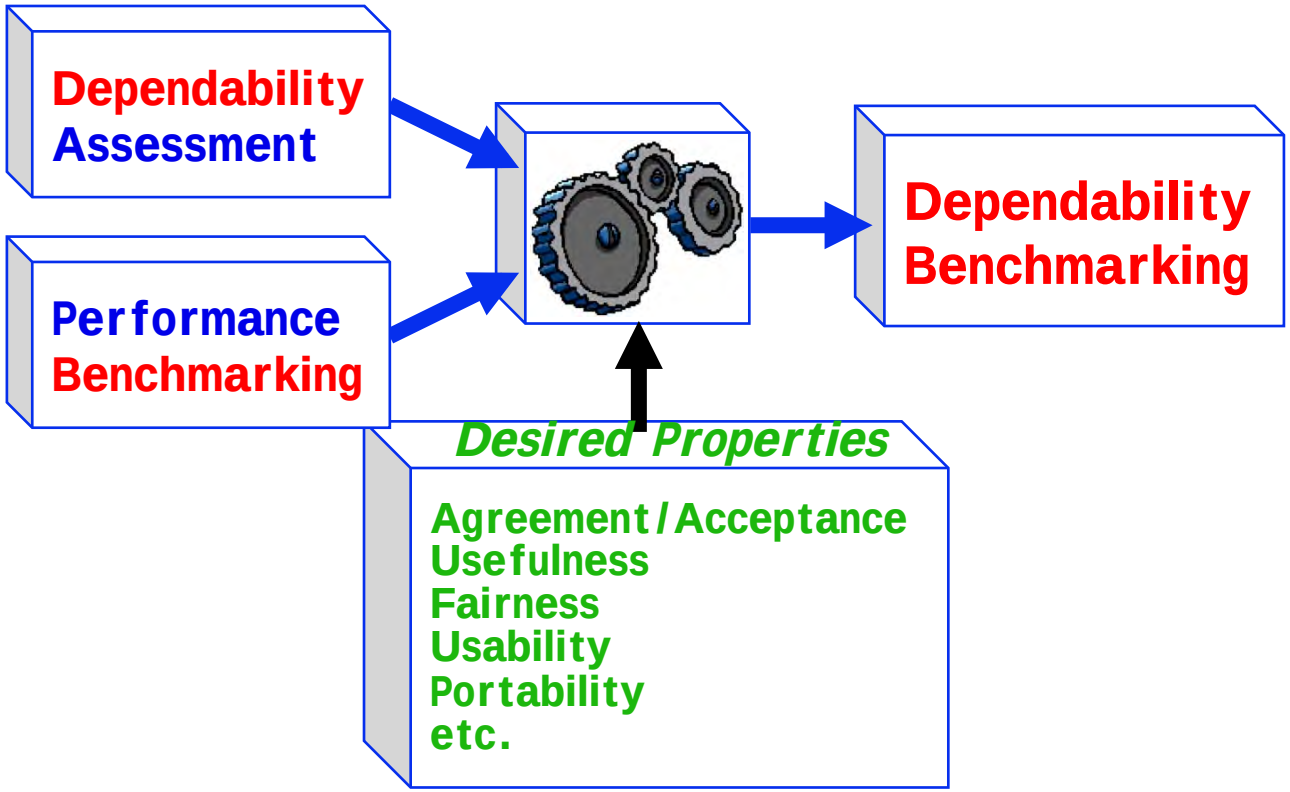


Views about Dependability Benchmarking

Naive View … :-)



More Realistic View:



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 procedure openly disclosed

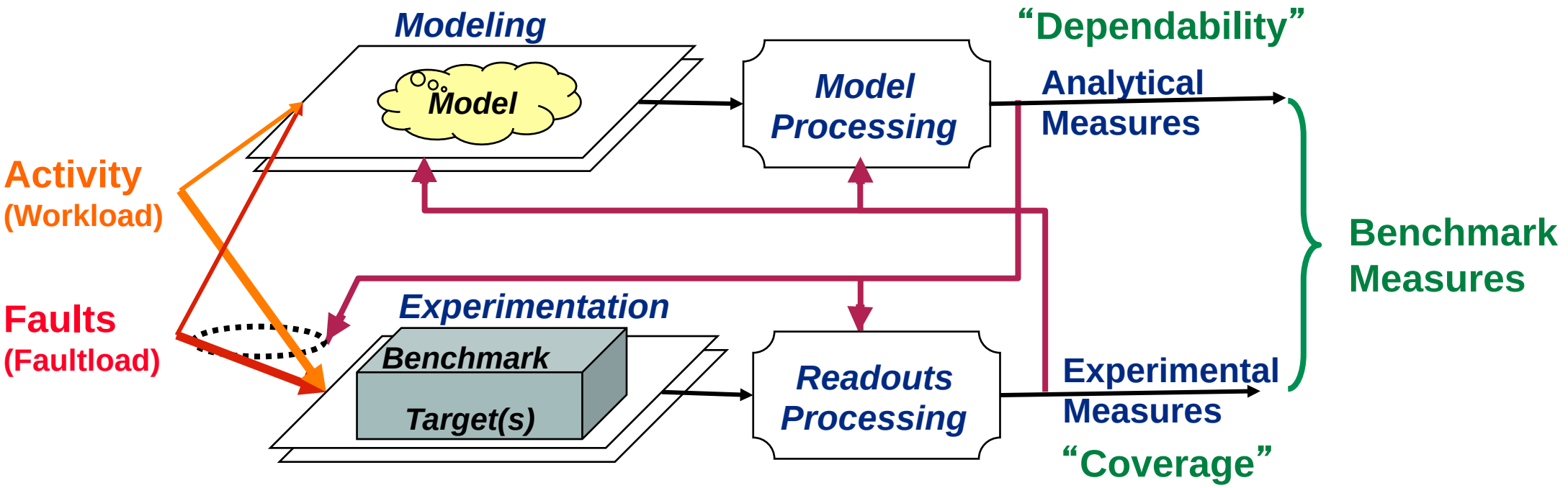
Common Properties

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

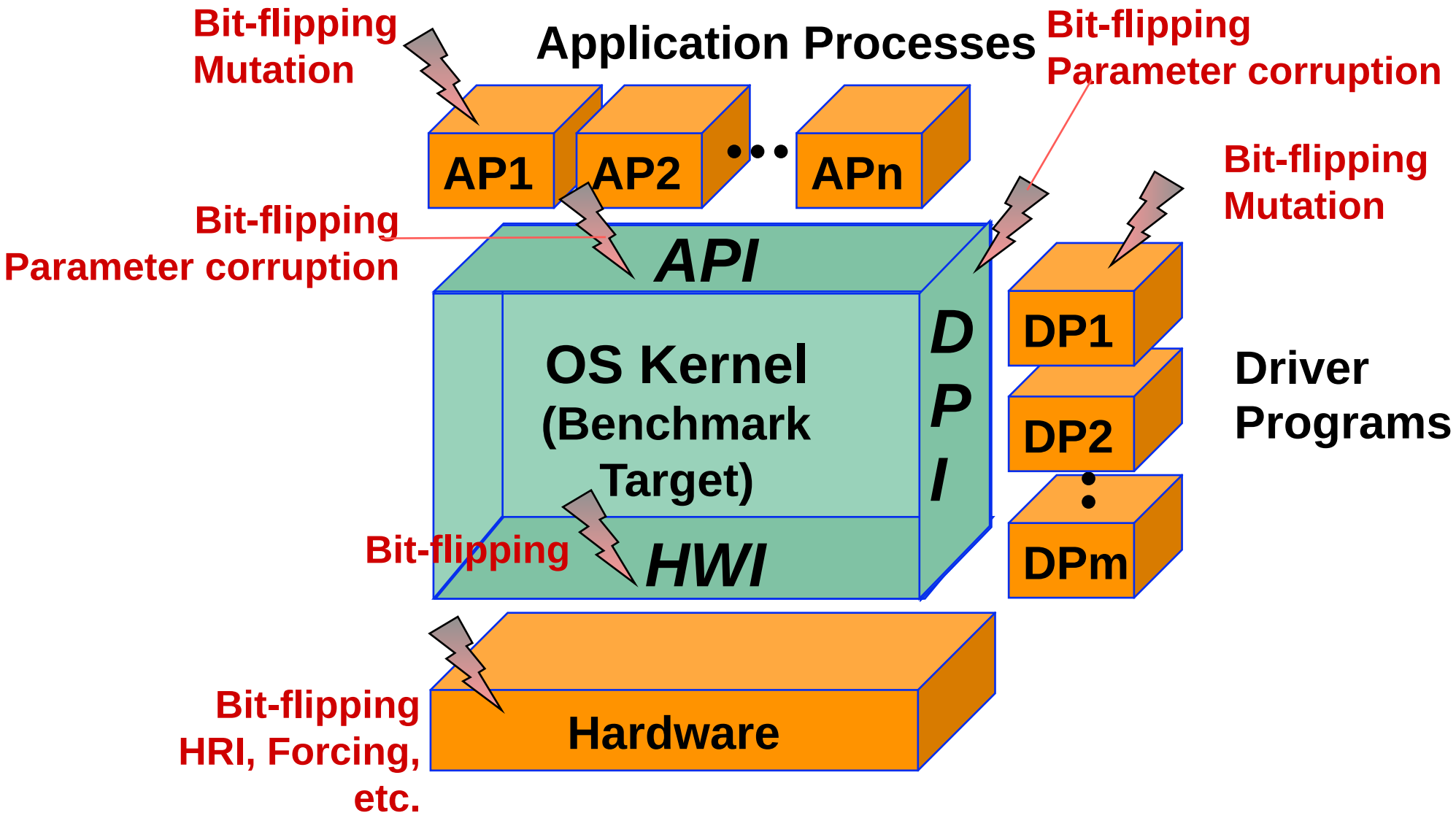


IST Project DBench (*Dependability Benchmarking*) — www.laas.fr/DBench and www.dbench.org

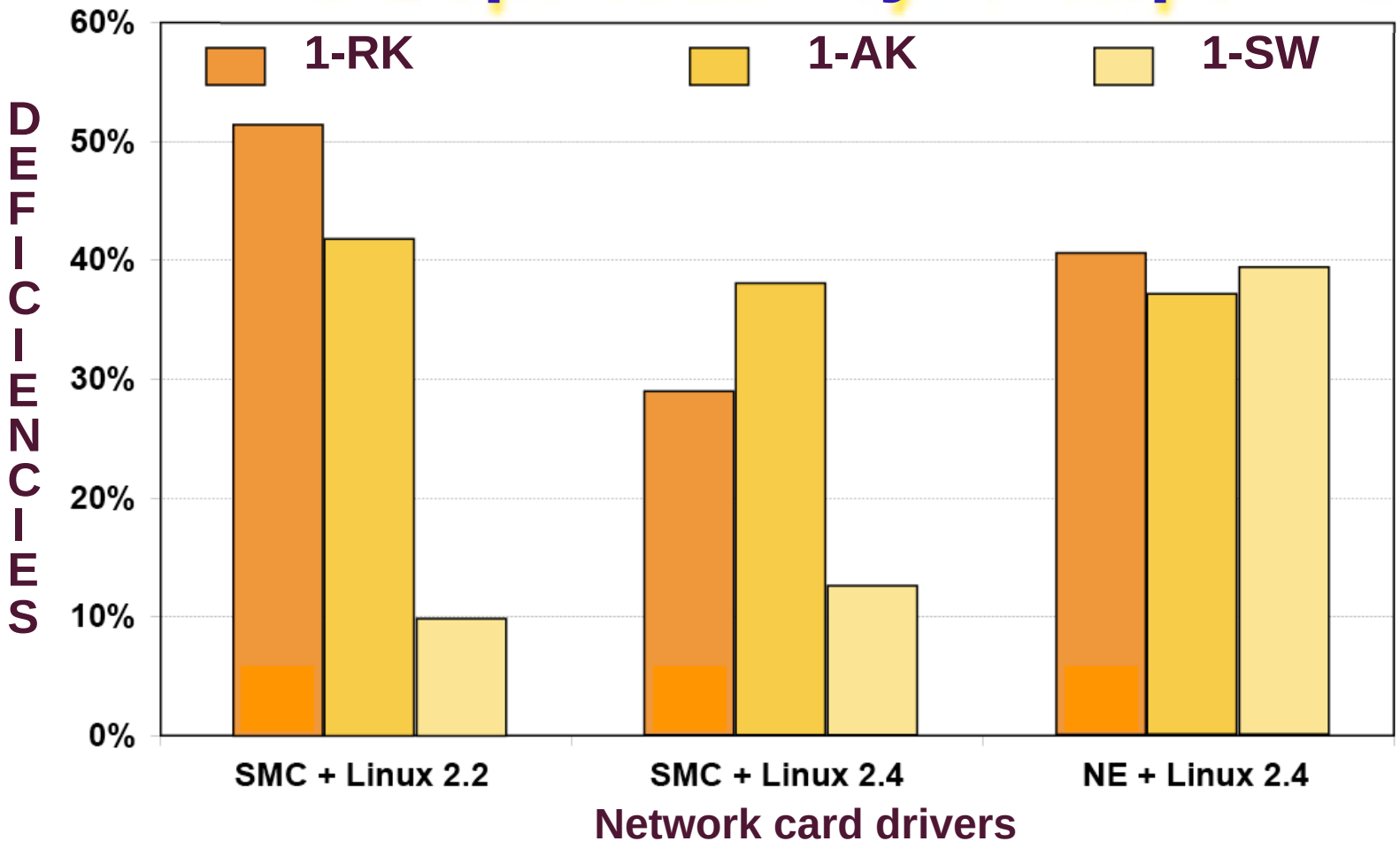
DBench DependabilityBenchmarking

—> **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



Linux

Kernel call: parameter corruption at DPI

ROCADE

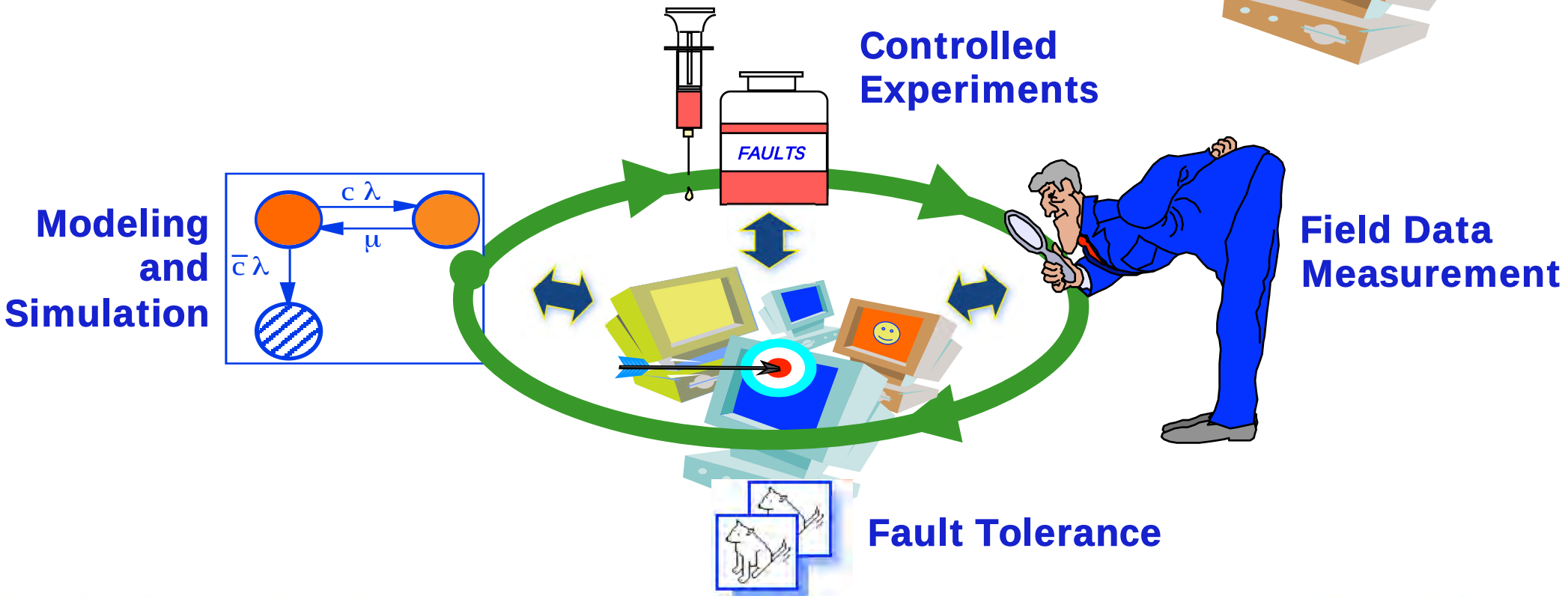
Robustnes Characterization Against Driver Errors

- **RK (Responsiveness of the Kernel)** = ↑ error notification
- **AK (Availability of the Kernel)** = ↓ kernel hangs
- **SW (Safety of the Workload)** = ↓ delivery of incorrect service

A.Albinet, J. Arlat, J.-C. Fabre
Benchmarking the Impact of Faulty Drivers: Applicationto the Linux Kernel
in *Dependability Benchmarking for Computer Systems* (K. Kanoun and L. Spainhower, Eds.), pp.285-310, IEEE CS Press and Wiley, 2008

Towards a Comprehensive Architecting and Assessment Framework

Emerging Features and Challenges



Mobility Configurability Target System … Highly evolutive Attacks

Merci !

Thanks!

Danke!

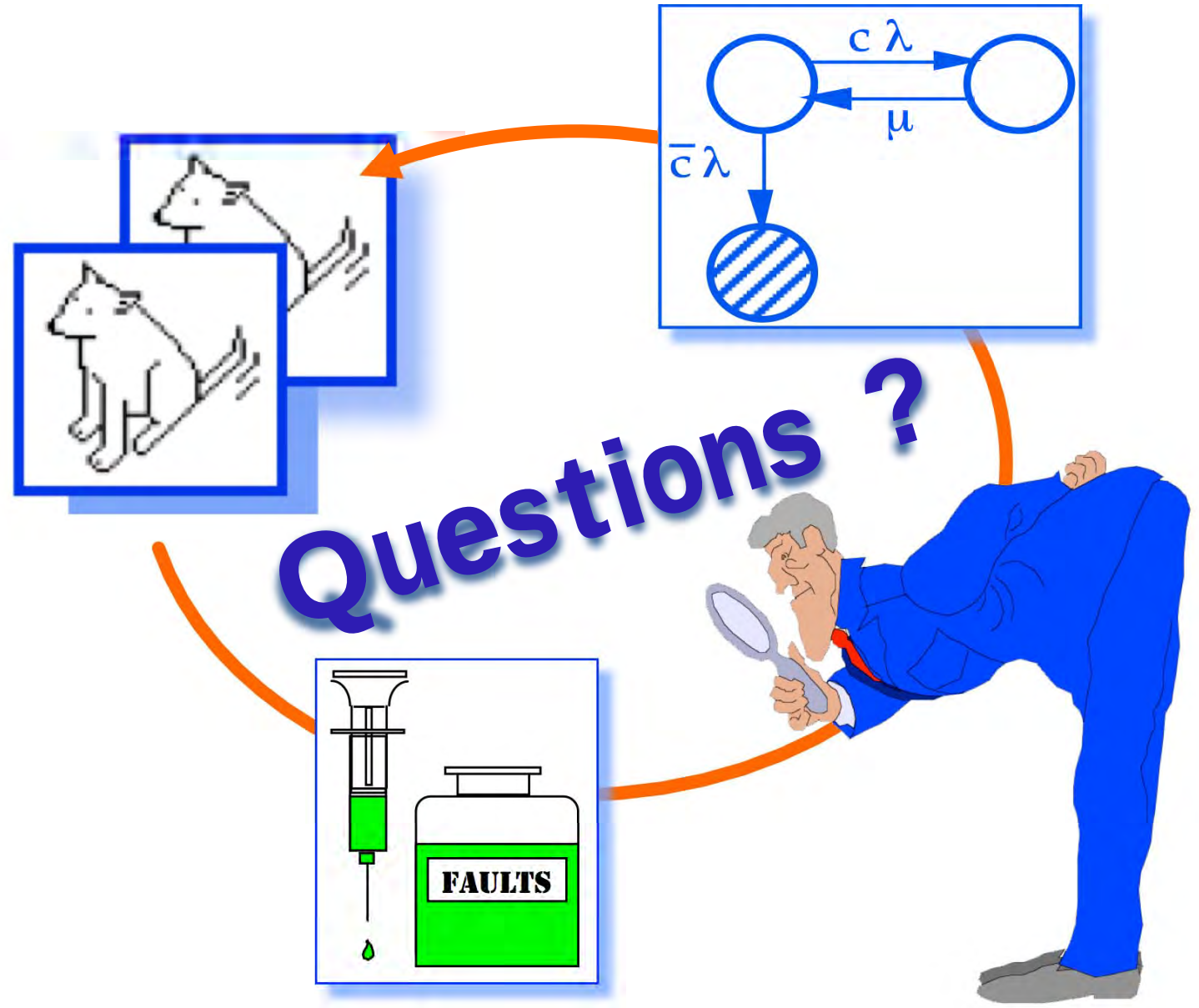
Gracias!

Grazie!

Obrigado!

ありがとう

謝謝


c λ
μ
c̄ λ
Questions ?
FAULTS

China Computer Federation — Fault-Tolerant Computing Committee

13th Conference on Fault-Tolerant Computing (CFTC-09)

Hailaer City, China — July, 20-21, 2009

[Souvenir from the first meeting with Prof. Yang]