Tutorial #1

Cross-Layer Soft-Error Resilience Analysis of Computing Systems

Alberto Bosio
École Centrale de Lyon, France

Stefano Di Carlo and **Alessandro Savino**
Politecnico di Torino, Italy

Dimitris Gizopoulos
University of Athens, Greece

Ramon Canal
Universitat Politècnica de Catalunya
Barcelona Supercomputing Center, Spain

CAL@DI

Computer Architecture Lab @ U Athens

POLITECNICO DI TORINO

TESTGROUP
TEST & RELIABILITY FOR NEXT TECHNOLOGIES

Part #1 – Introduction and basic concepts

Stefano Di Carlo

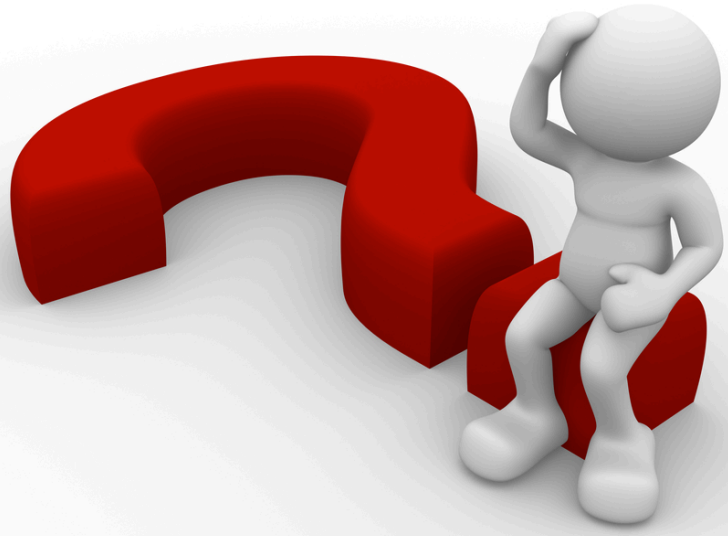
DSN 2020
50TH IEEE/IFIP INT. CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS

GOAL

Cross-Layer Soft-Error Resilience of Computing Systems

WHAT is it?

WHY do we need it?



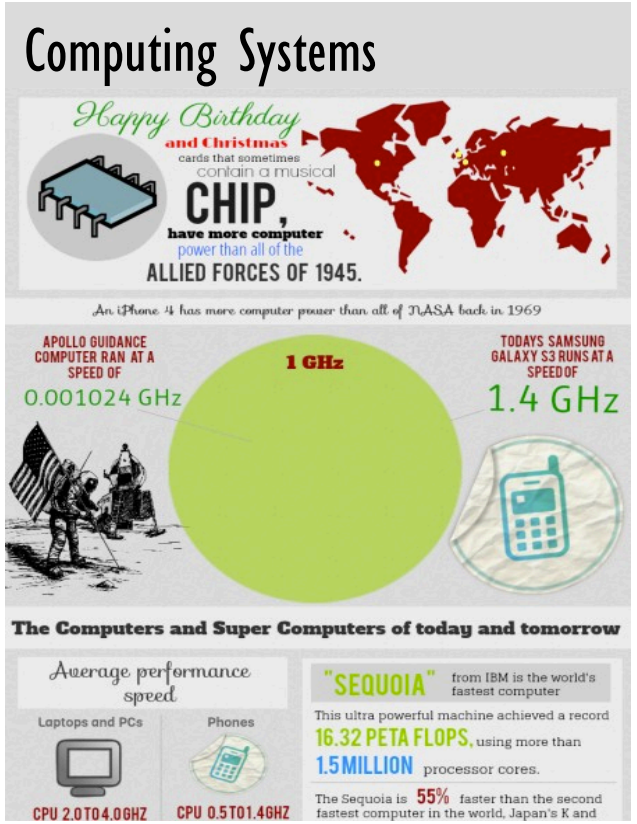
HOW do we obtain it?

COMPUTING SYSTEMS

Fundamental system properties

FUNCTIONALITY

SIZE, WEIGHT and POWER (SWaP)



PERFORMANCE

DEPENDABILITY

COSTS (design, manufacturing)

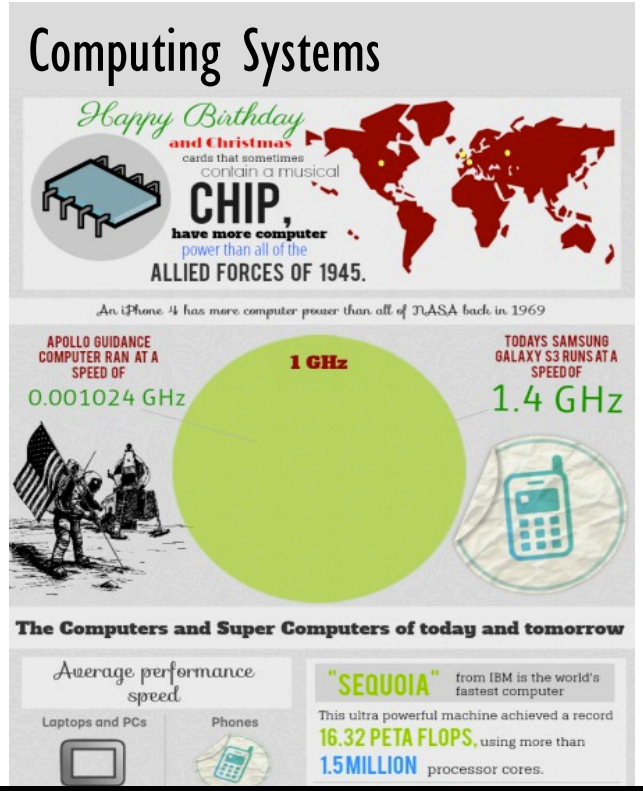
SECURITY

COMPUTING SYSTEMS

Fundamental system properties

FUNCTIONALITY

SIZE, WEIGHT and POWER (SWaP)



PERFORMANCE

DEPENDABILITY

COSTS

All properties should be considered when designing a computing system

DEPENDABILITY

General definitions

Dependability of a system is the ability to avoid **service failures** that are more frequent and more severe than is acceptable

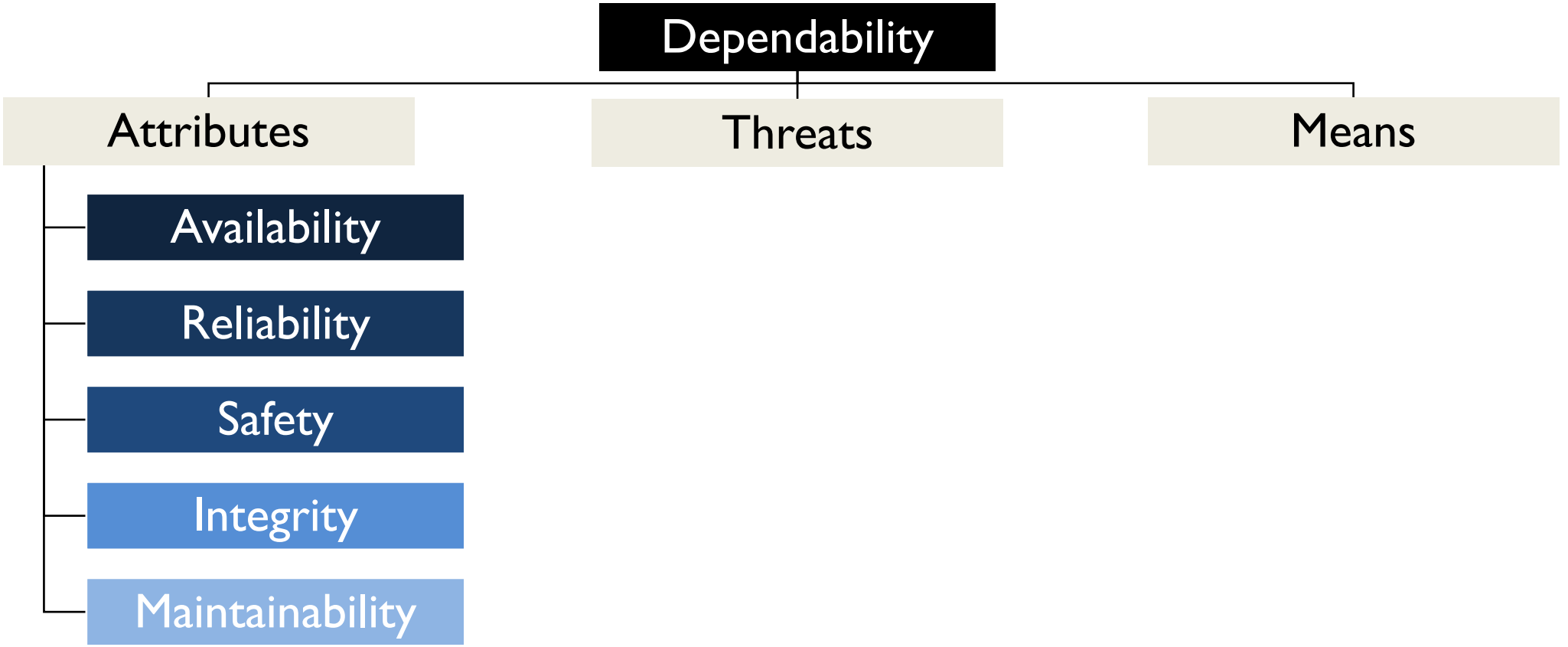
Dependability is the ability to deliver **service** that can justifiably be trusted

[Avizienis et al., IEEE TDSC 2004]

1980, joint committee on "Fundamental Concepts and Terminology" formed by the TC on Fault-Tolerant Computing of the IEEE CS and the IFIP WG 10.4 "Dependable Computing and Fault Tolerance."

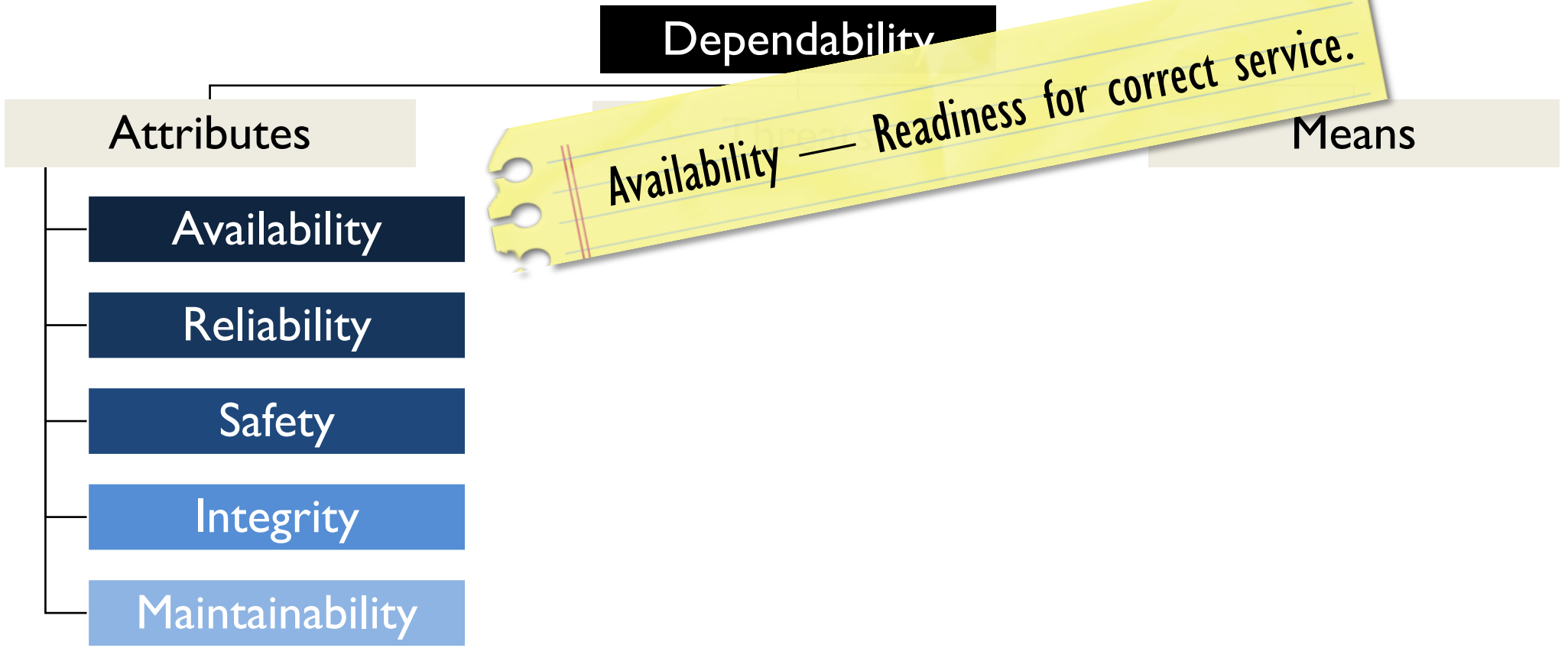
DEPENDABILITY

Specific concepts



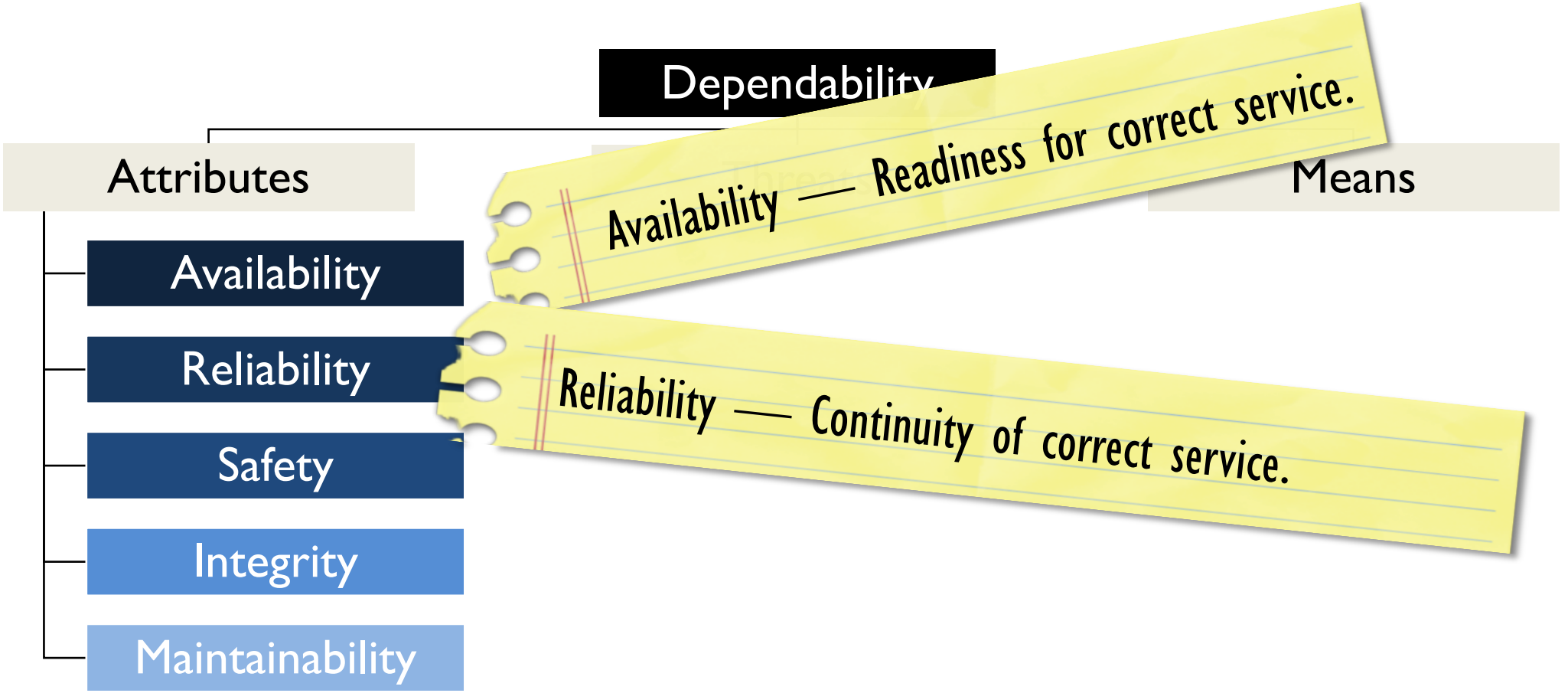
DEPENDABILITY

Specific concepts



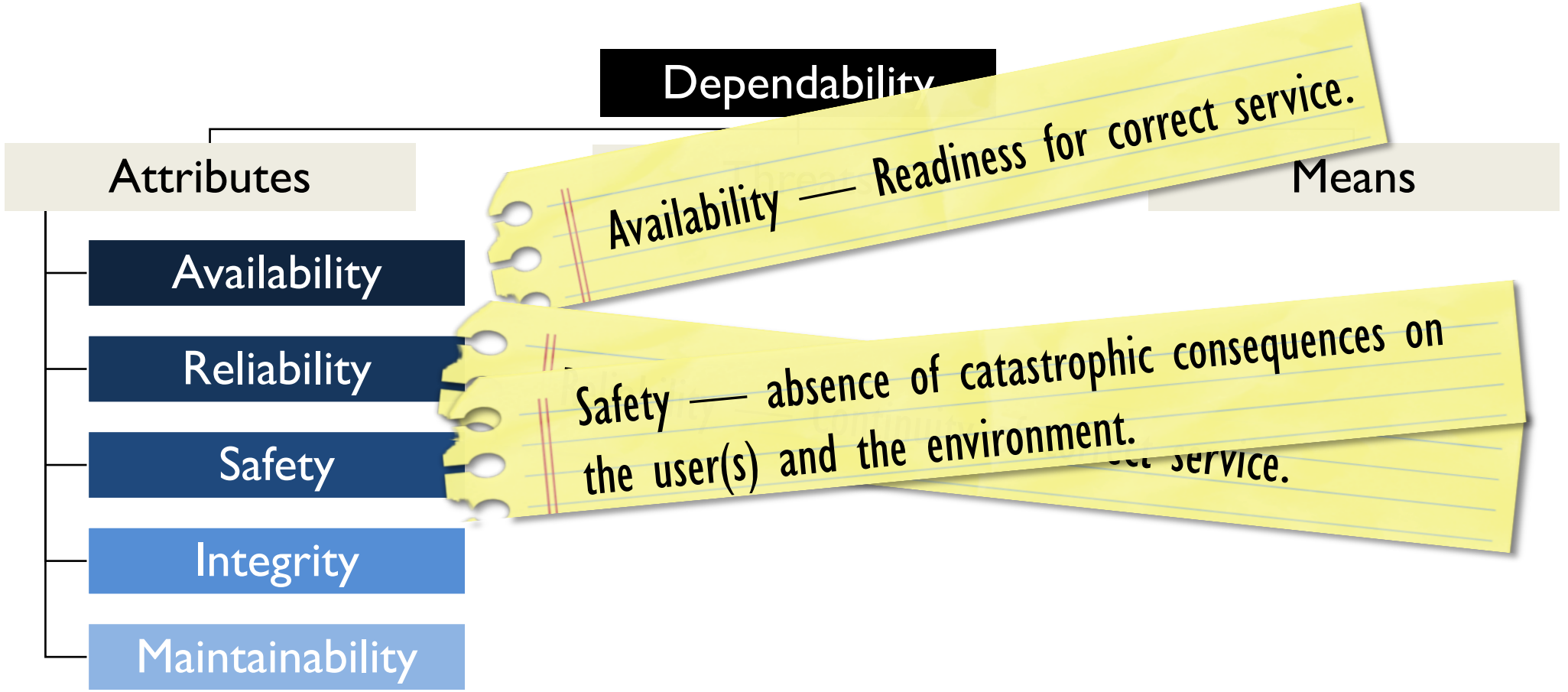
DEPENDABILITY

Specific concepts



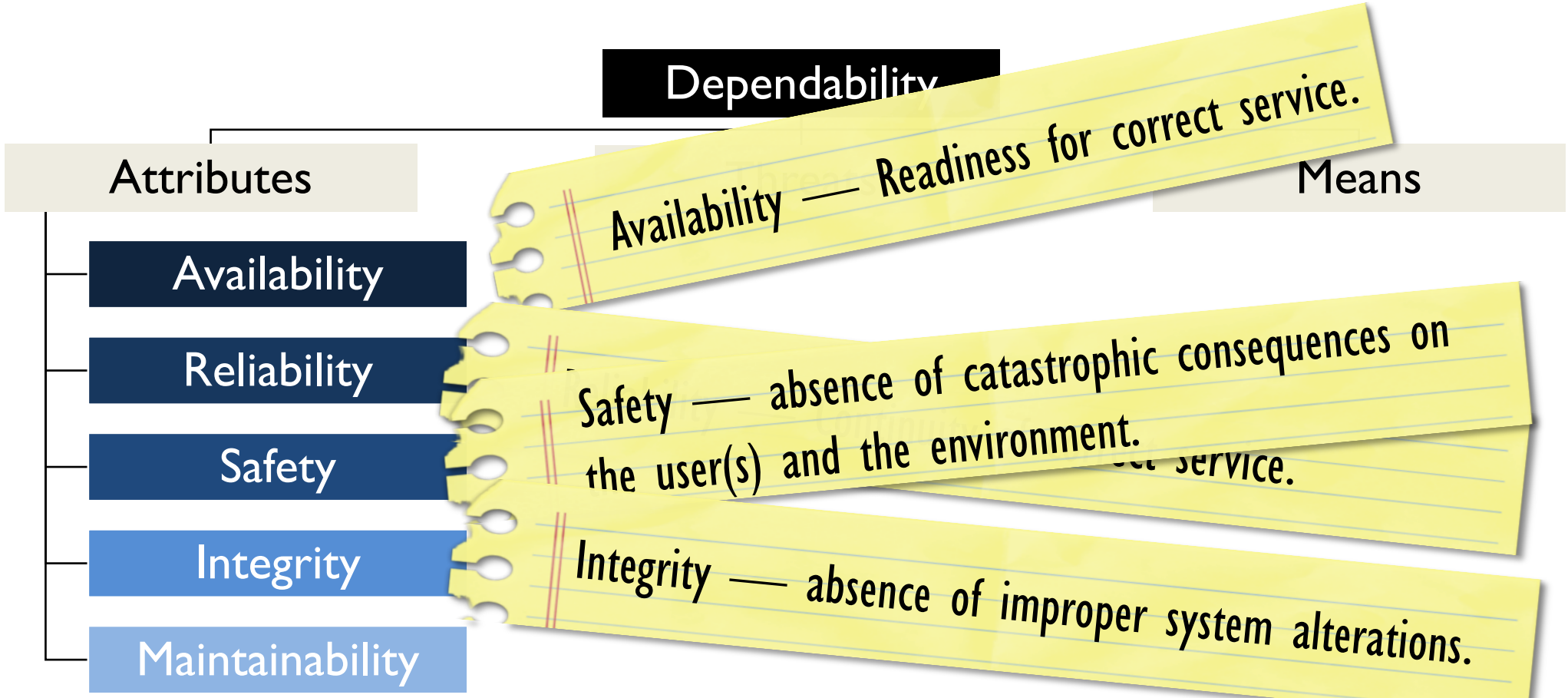
DEPENDABILITY

Specific concepts



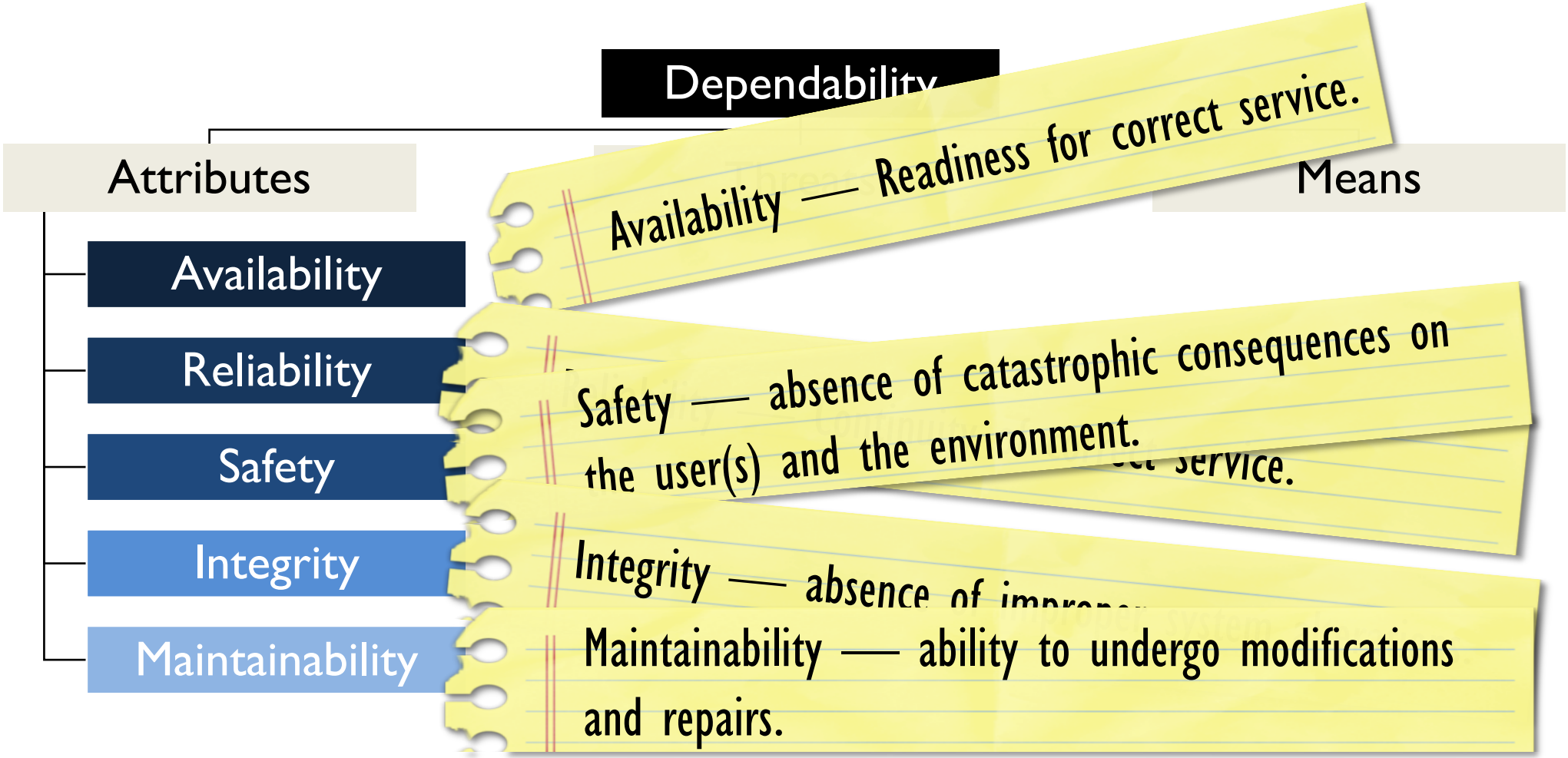
DEPENDABILITY

Specific concepts



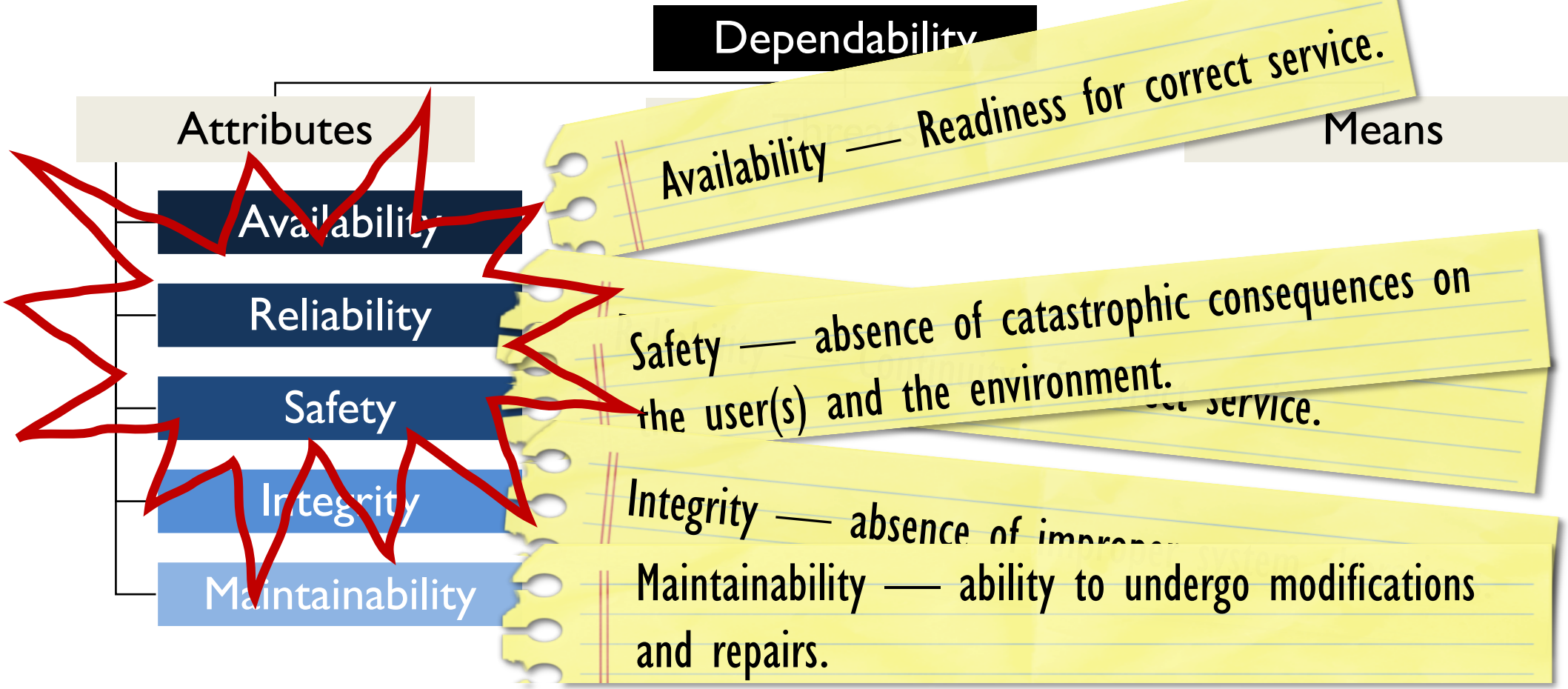
DEPENDABILITY

Specific concepts



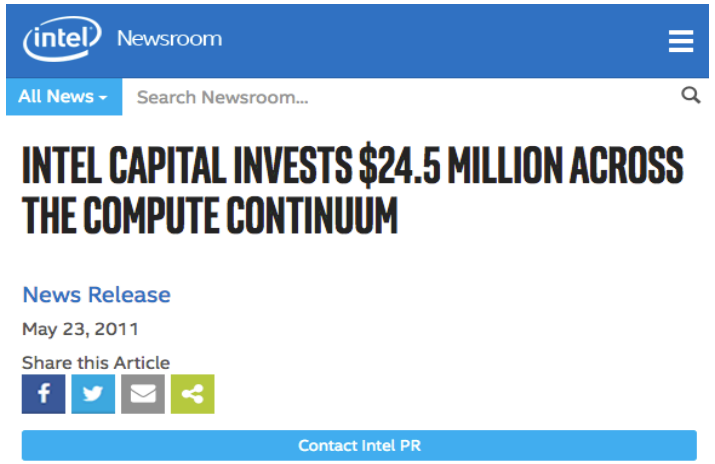
DEPENDABILITY

Specific concepts



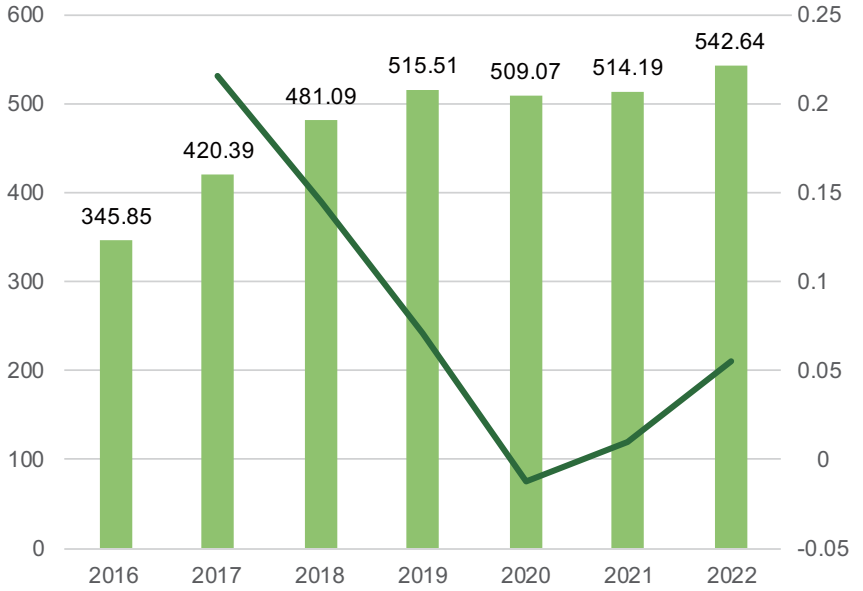
RELIABILITY IMPORTANCE

The concept of Computing Continuum

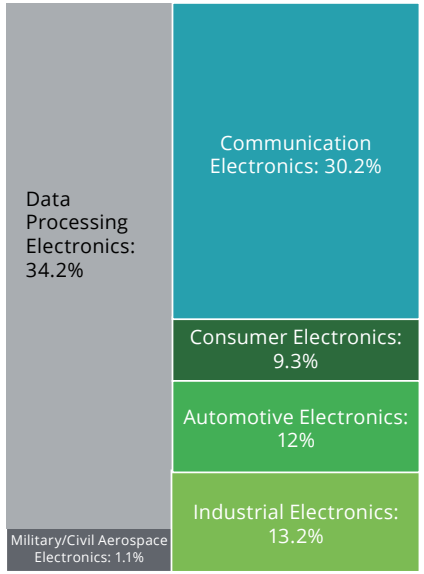


[https://newsroom.intel.com/news-releases/intel-capital-invests-24-5-million-across-the-compute-continuum/]

Figure: Global semiconductor sales revenue (2016-2022, billion USD)



542.64 billion (2022)



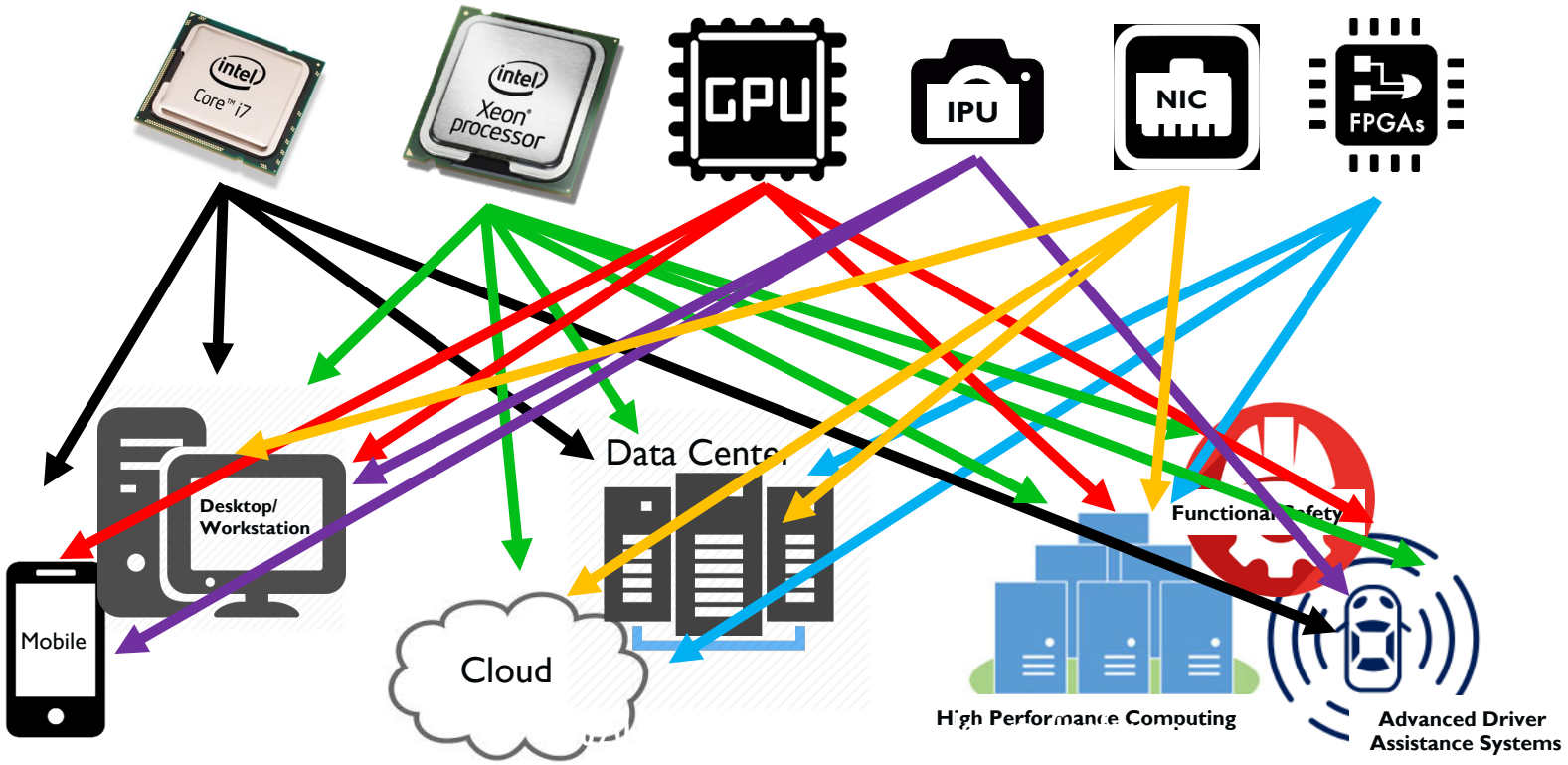
https://www2.deloitte.com/content/dam/Deloitte/cn/Documents/technology-media-telecommunications/deloitte-cn-tmt-semiconductors-the-next-wave-en-190422.pdf

Intel's compute continuum

Today's computing is a true continuum that ranges from smartphones to mission-critical datacenter machines, and from desktops to automobiles.

THE COMPUTE CONTINUUM

Same technologies and architectures across markets & designs



[Arijit Biswas Intel, SELSE 2018]

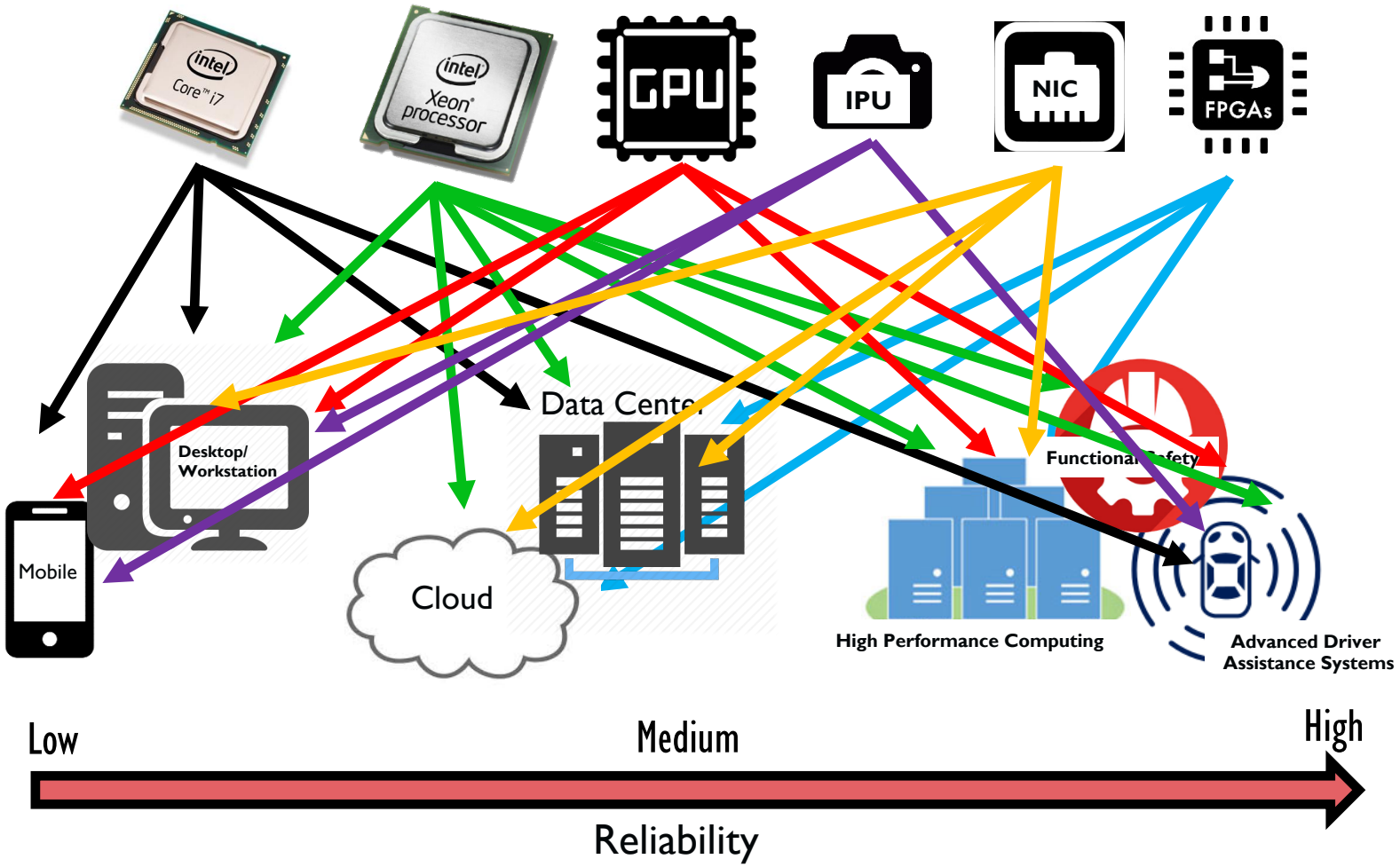
THE COMPUTE CONTINUUM

Same technologies and architectures across markets & designs

[Arijit Biswas Intel, SELSE 2018]

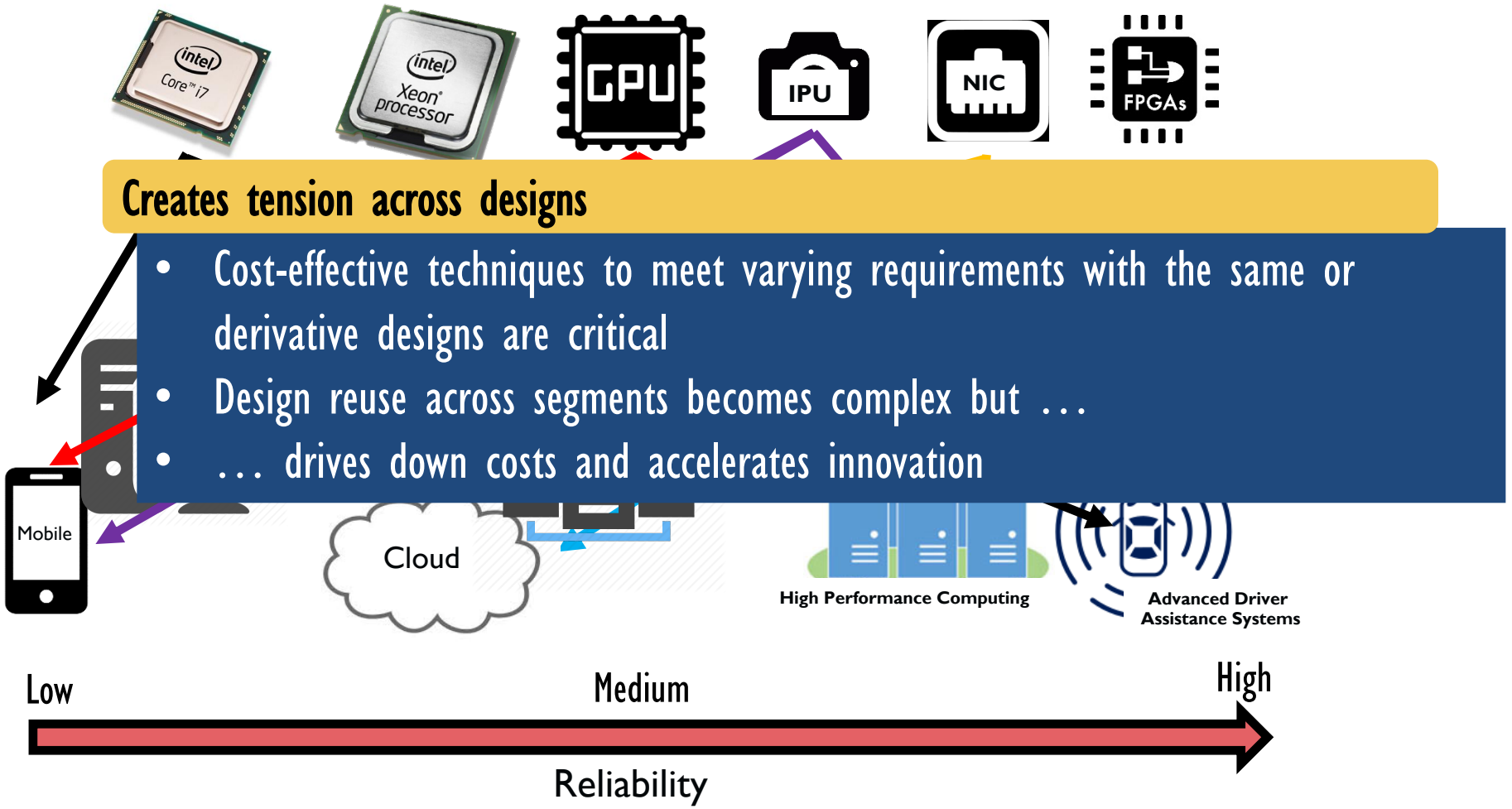
THE COMPUTE CONTINUUM

Reliability requirements vary across markets & designs



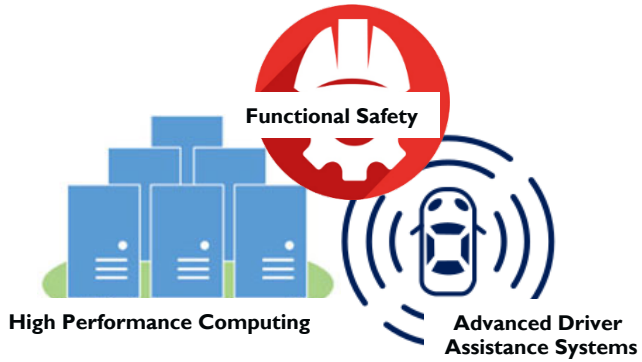
THE COMPUTE CONTINUUM

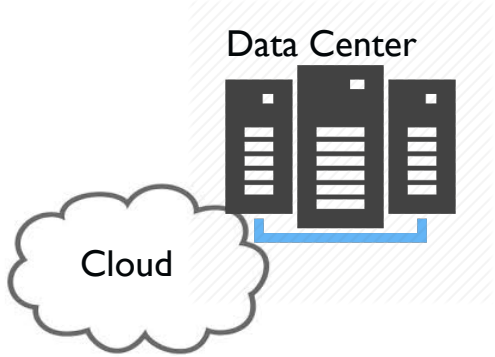
Reliability requirements vary across markets & designs



THE COMPUTE CONTINUUM

Impact of failures on human society





High

- Direct impact to human life/safety critical
- Wasted time/effort, indirect impact to human life

Medium

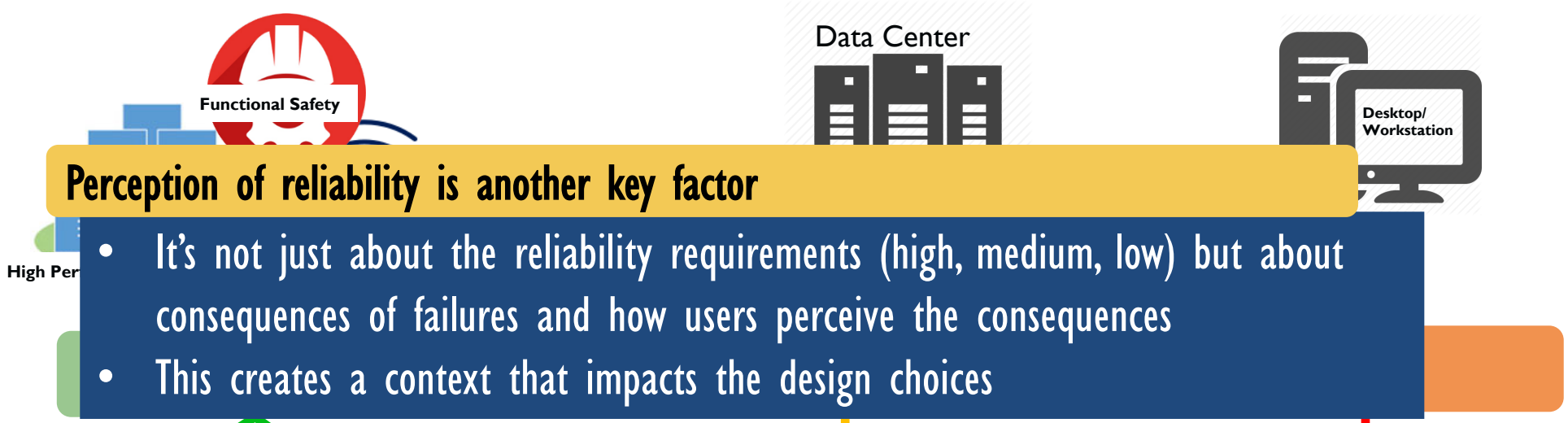
- From dropped frames on a streaming movie to corrupt bank account balances…can have real financial impact

Low

- Depends on usage, probably not life or livelihood threatening

THE COMPUTE CONTINUUM

Impact of failures on human society



Perception of reliability is another key factor

- It's not just about the reliability requirements (high, medium, low) but about consequences of failures and how users perceive the consequences
- This creates a context that impacts the design choices

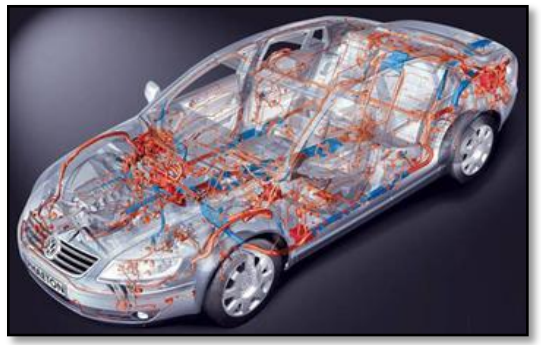
Direct impact to human life/safety critical	Wasted time/effort, indirect impact to human life	From dropped frames on a streaming movie to corrupt bank account balances…can have real financial impact	Depends on usage, probably not life or livelihood threatening

THE COMPUTE CONTINUUM

The main reliability drivers

Driven by performance

Autonomous driving

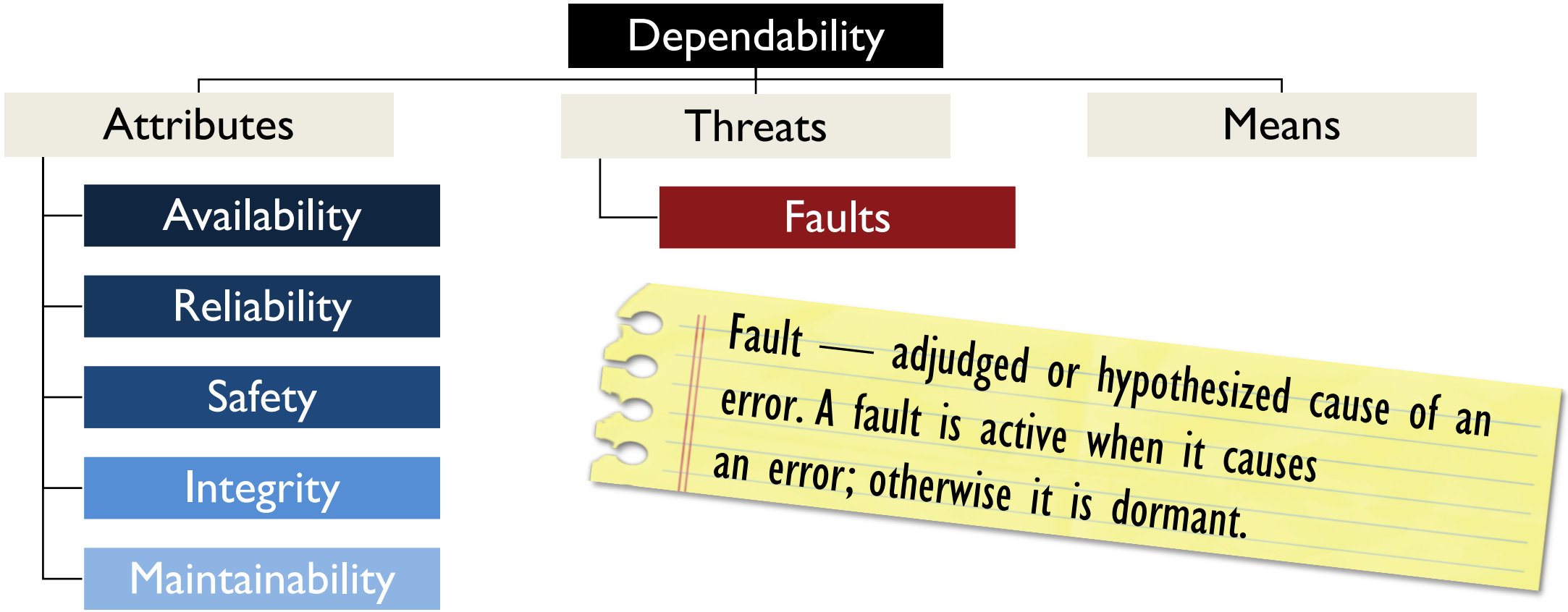


Driven by Safety Requirements

- Drive to Exascale (10^{18} FLOPS) performance results in problems due to sheer scale
- Failure is inevitable – need to ensure sufficient fault tolerance capabilities to get enough work done to make it worthwhile
- Higher reliability per Node enables More Nodes = More Performance
- Goal is fully autonomous self-driving vehicles
- Requires high performance & high reliability
- Reliability requirements established by industry specifications on Functional Safety
- Different parts of the data flow can have different reliability requirements

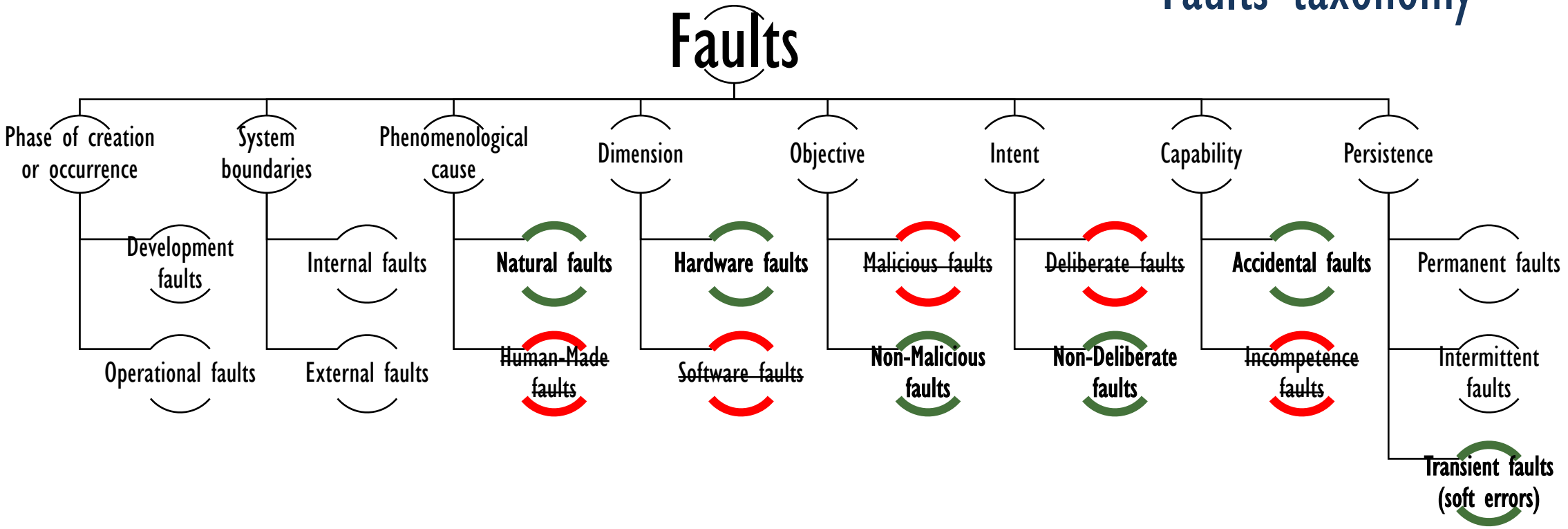
DEPENDABILITY

Specific concepts



DEPENDABILITY

Faults taxonomy



DEPENDABILITY

Faults taxonomy

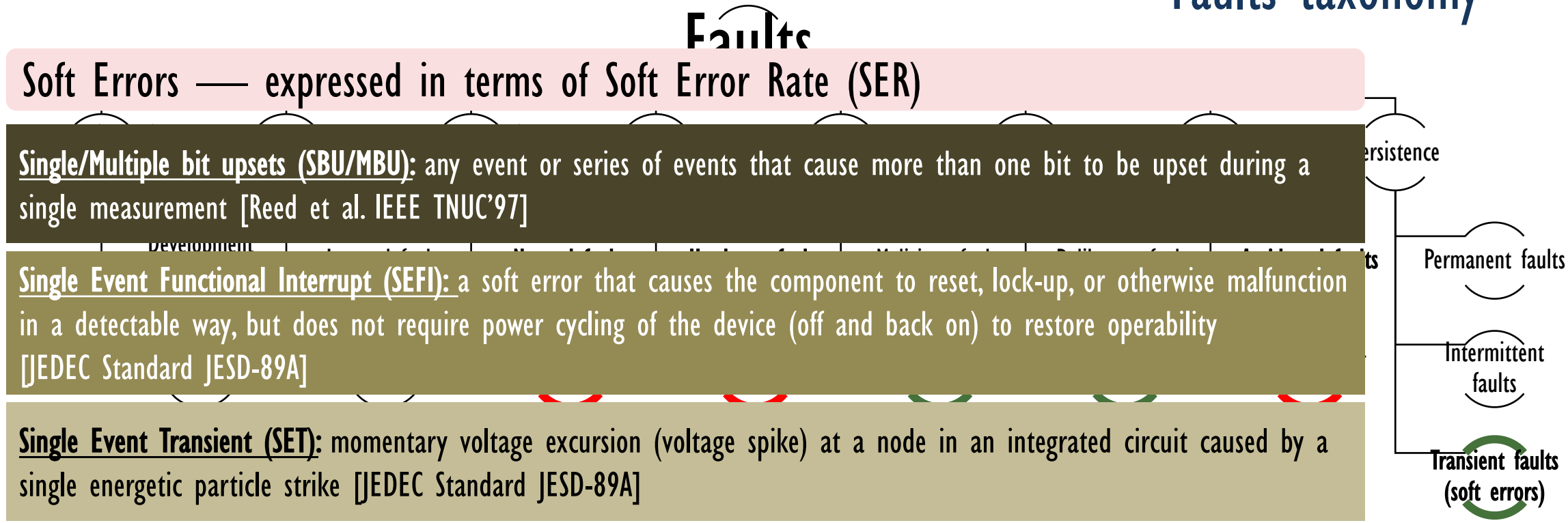
Soft Errors — expressed in terms of Soft Error Rate (SER)

Single/Multiple bit upsets (SBU/MBU): any event or series of events that cause more than one bit to be upset during a single measurement [Reed et al. IEEE TNUC'97]

Single Event Functional Interrupt (SEFI): a soft error that causes the component to reset, lock-up, or otherwise malfunction in a detectable way, but does not require power cycling of the device (off and back on) to restore operability [JEDEC Standard JESD-89A]

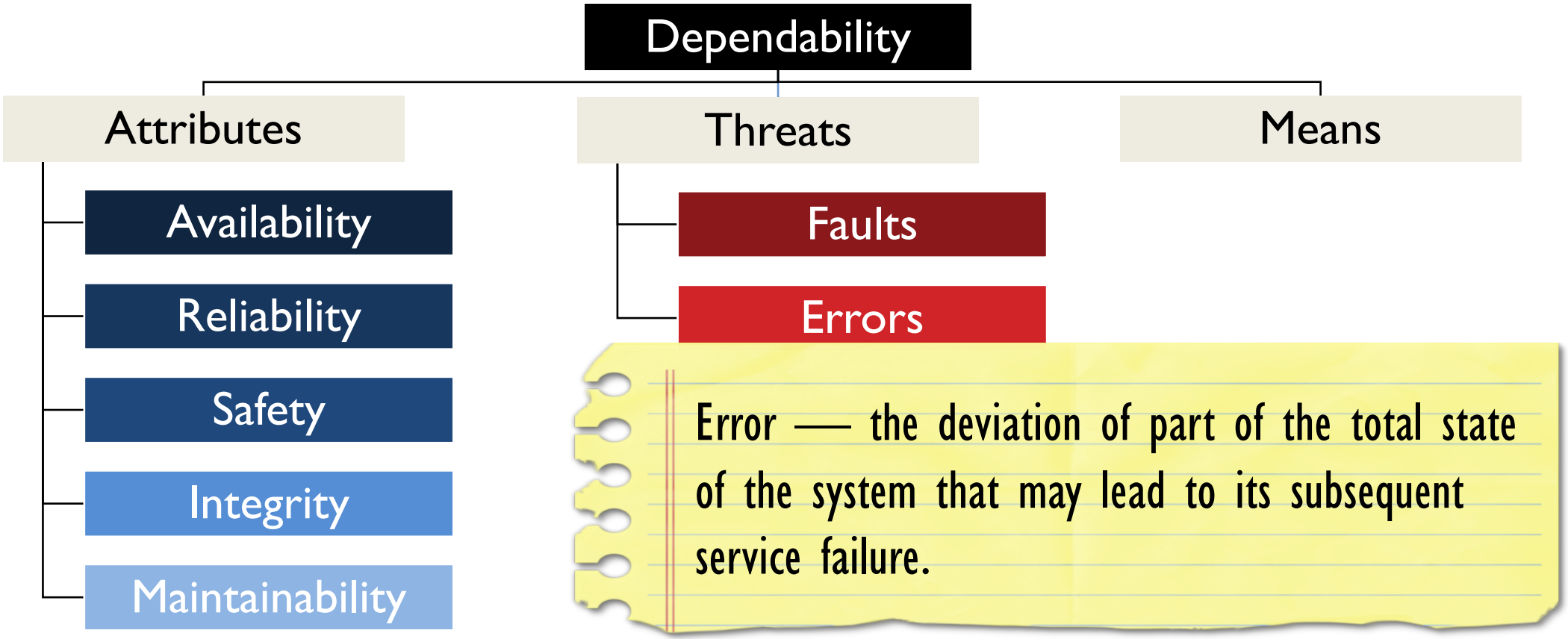
Single Event Transient (SET): momentary voltage excursion (voltage spike) at a node in an integrated circuit caused by a single energetic particle strike [JEDEC Standard JESD-89A]

Single Event Latch-Up (SEL): abnormal high-current state in a device caused by the passage of a single energetic particle through sensitive regions of the device structure and resulting in the loss of device functionality [JEDEC Standard JESD-89A]



DEPENDABILITY

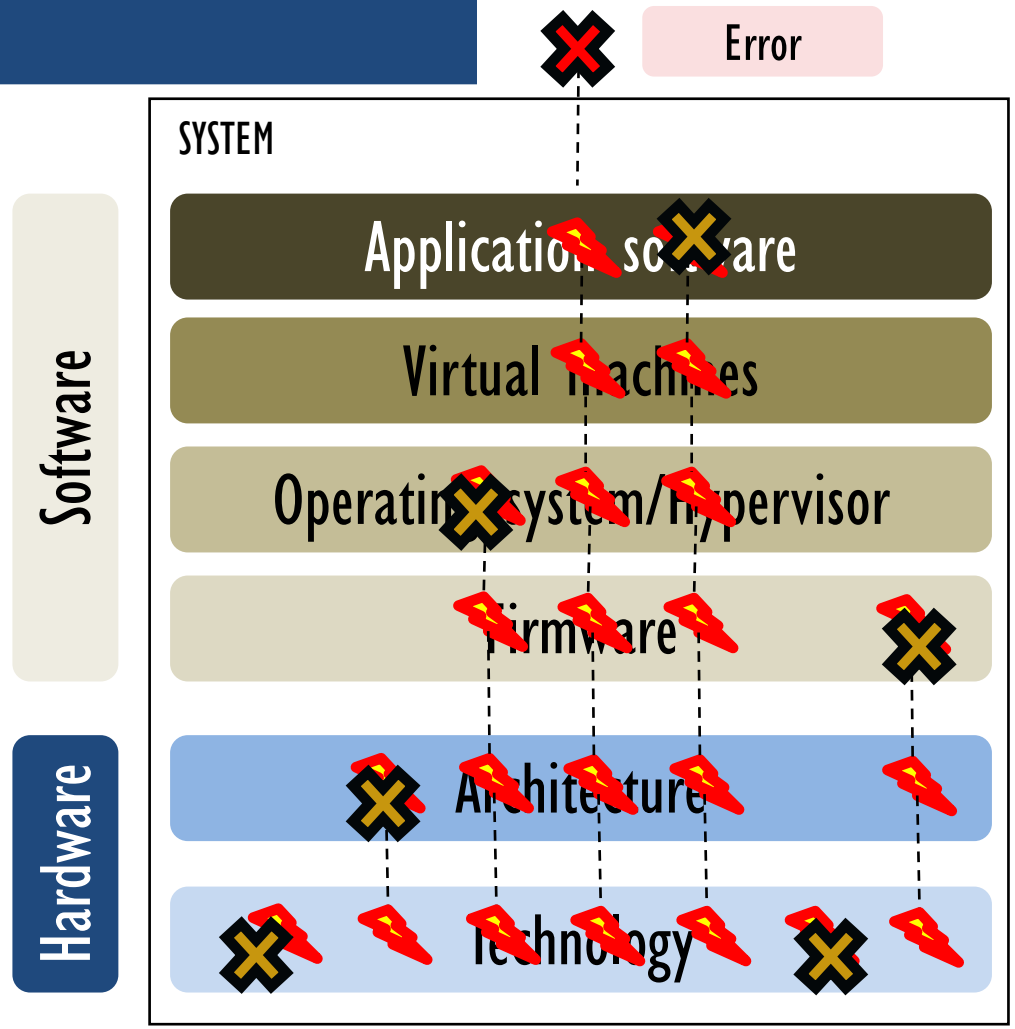
Specific concepts



CROSS-LAYER RESILIENCE

The system vulnerability stack *[Sridharan & Kaeli, ISCA'10]*

Faults can be naturally filtered at various levels of the system stack.



Architecture Vulnerability Factor (AVF)

Program Vulnerability Factor (PVF)

Virtual Machine Vulnerability Factor (VMVF)

Operating System Vulnerability Factor (OSVF)

[Vilas&Kaeli ISCA'10]

Firmware Vulnerability Factor (FVF)

Hardware Vulnerability Factor (HVF) [Vilas&Kaeli ISCA'10]

Time Vulnerability Factor (TVF) [Seifert et al. TDMR'04]

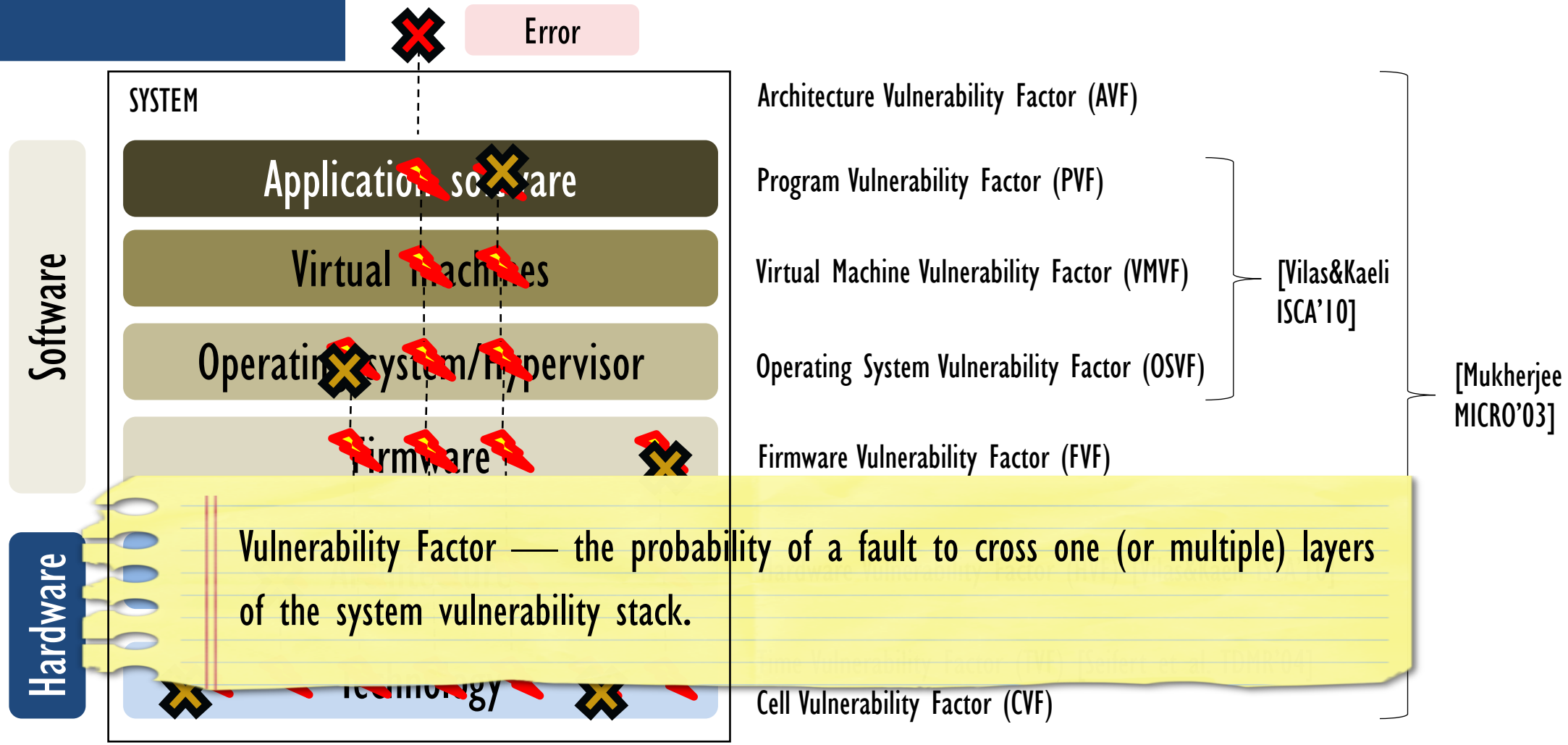
Cell Vulnerability Factor (CVF)

[Mukherjee MICRO'03]

CROSS-LAYER RESILIENCE

The system vulnerability stack *[Sridharan & Kaeli, ISCA'10]*

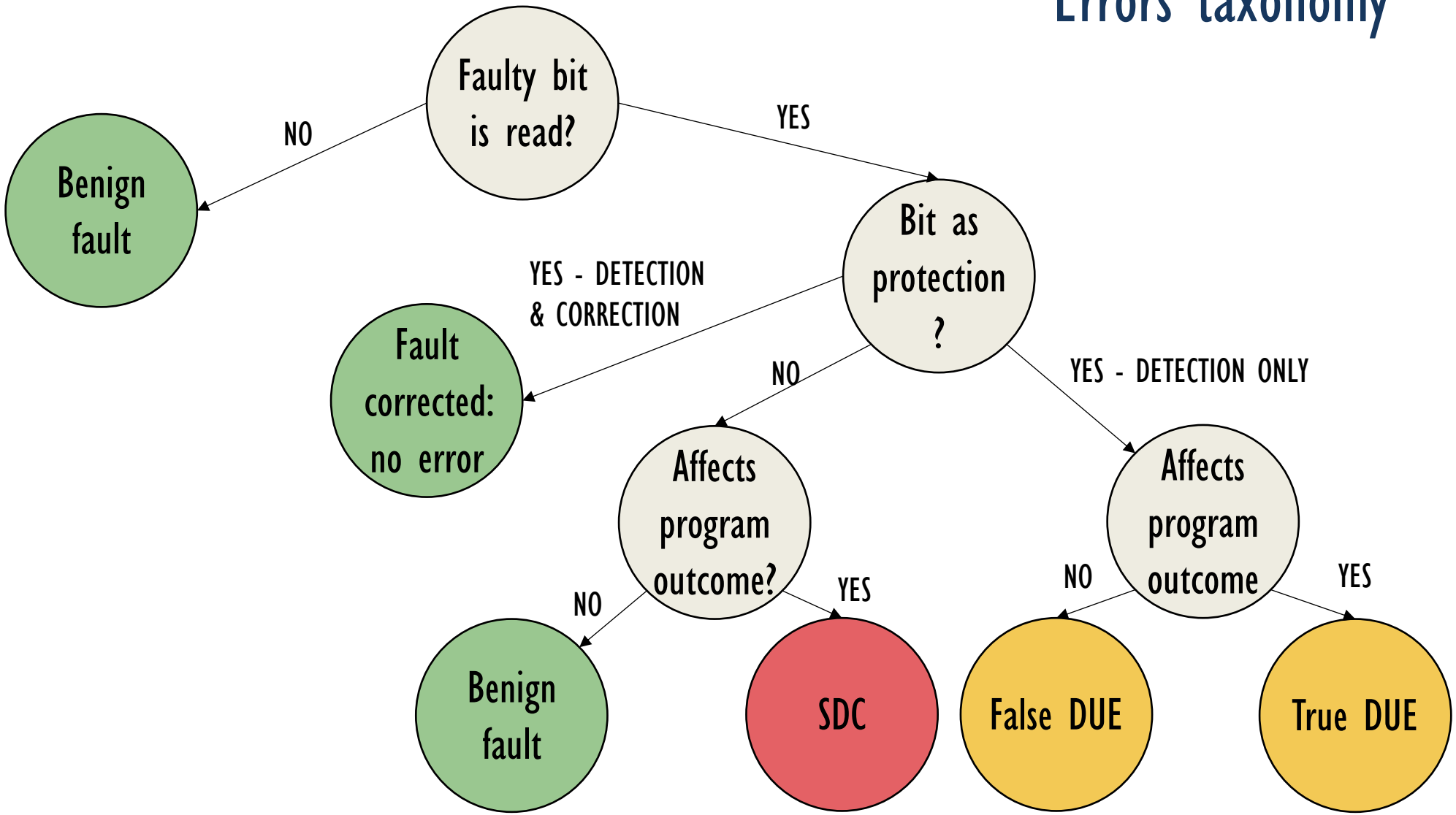
Faults can be naturally filtered at various levels of the system stack.



Vulnerability Factor — the probability of a fault to cross one (or multiple) layers of the system vulnerability stack.

DEPENDABILITY

Errors taxonomy

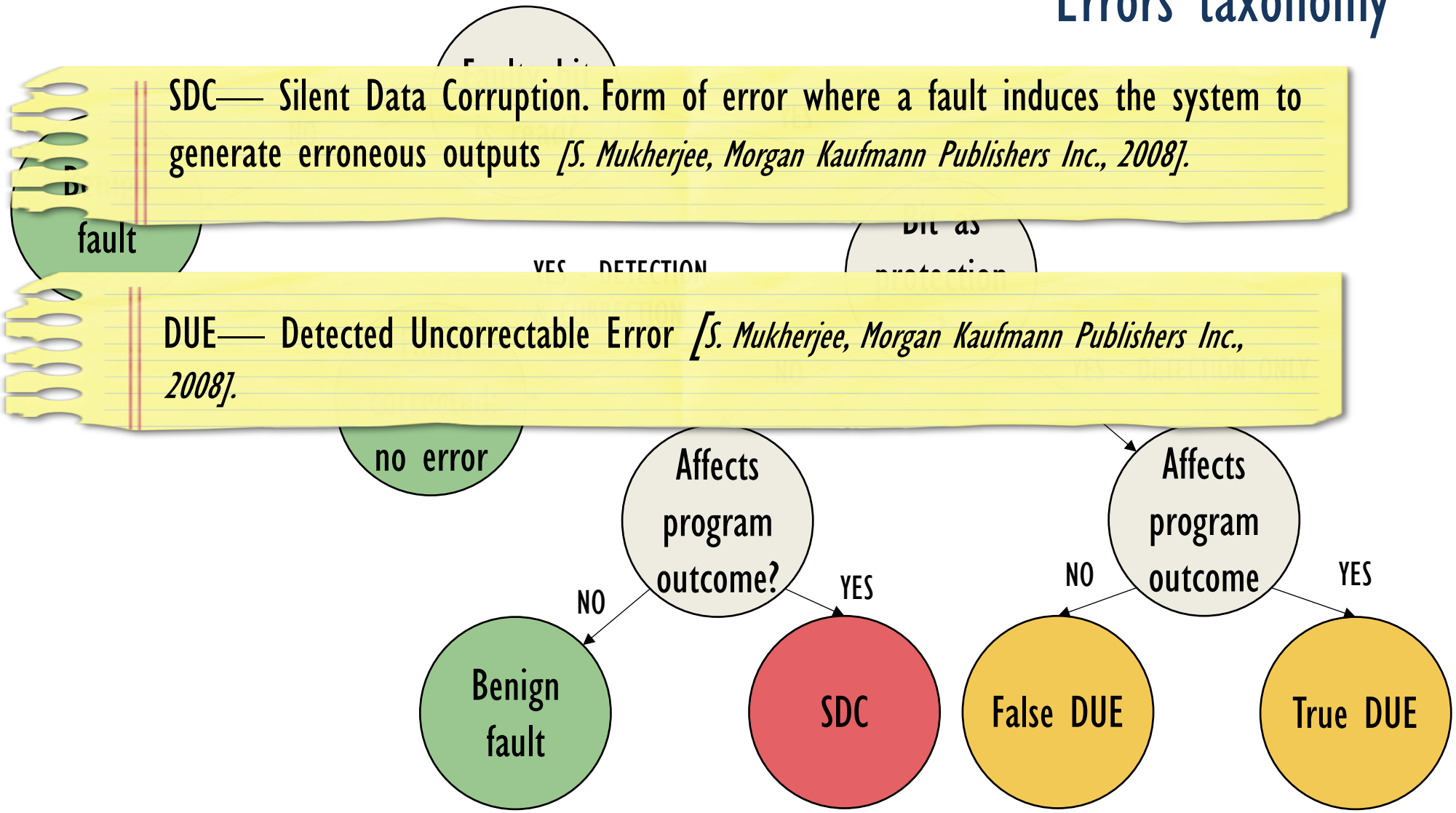


DEPENDABILITY

Errors taxonomy

SDC— Silent Data Corruption. Form of error where a fault induces the system to generate erroneous outputs *[S. Mukherjee, Morgan Kaufmann Publishers Inc., 2008].*

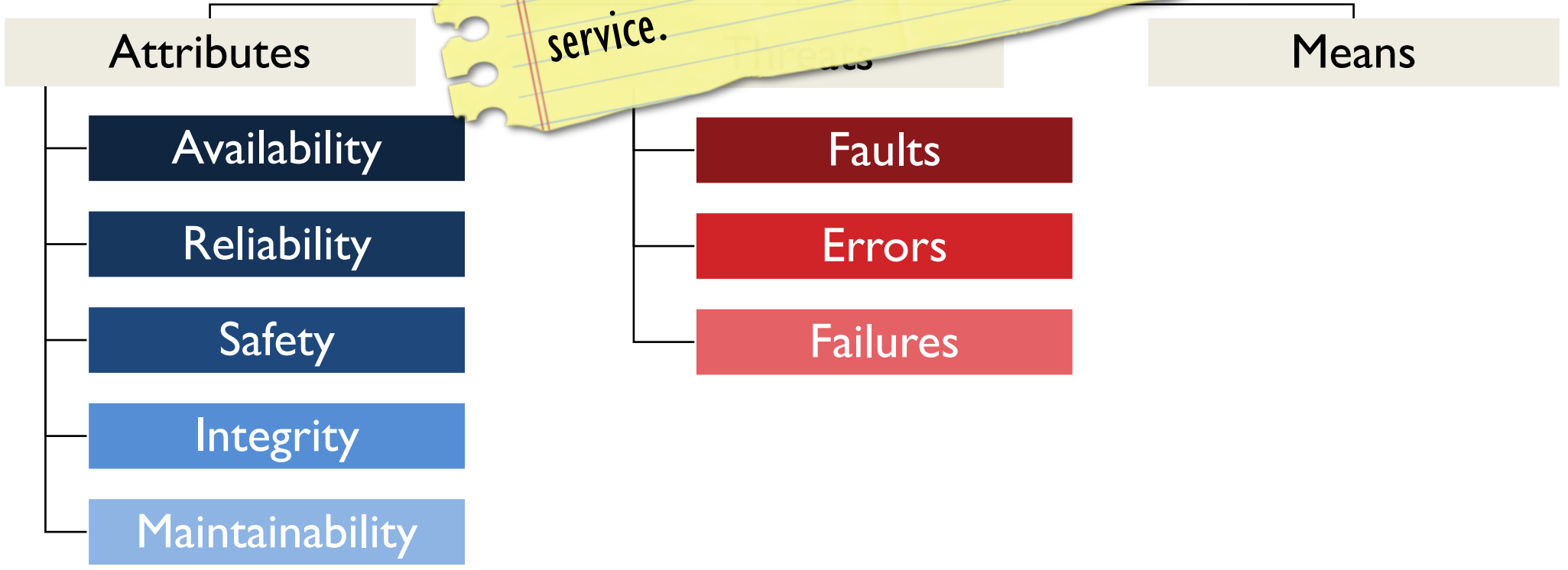
DUE— Detected Uncorrectable Error *[S. Mukherjee, Morgan Kaufmann Publishers Inc., 2008].*



DEPENDABILITY

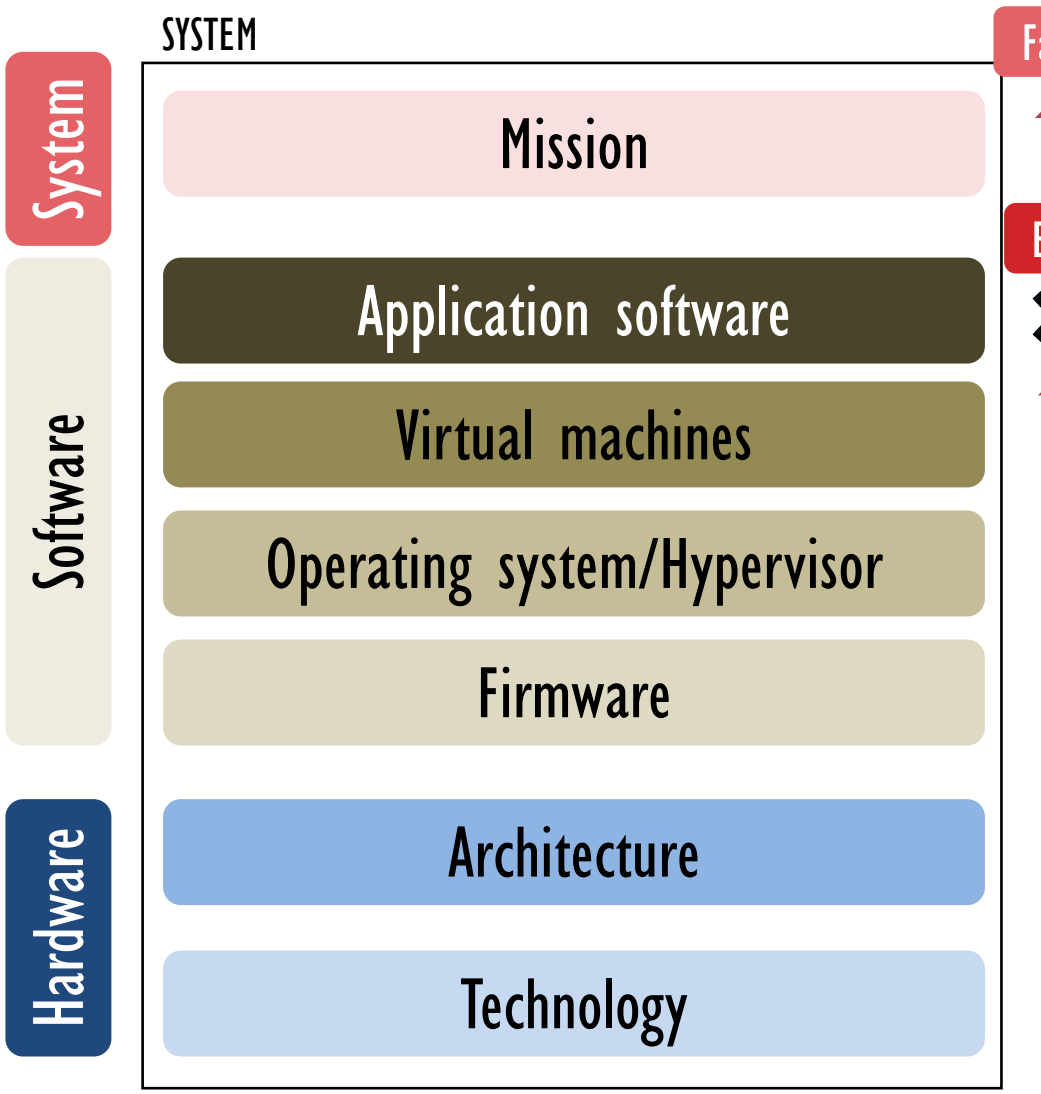
ific concepts

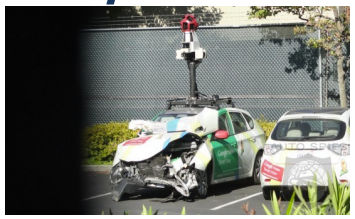
Failure — an event that occurs when the delivered service deviates from correct service.



CROSS-LAYER RESILIENCE

The system vulnerability stack *[Sridharan & Kaeli, ISCA'10]*





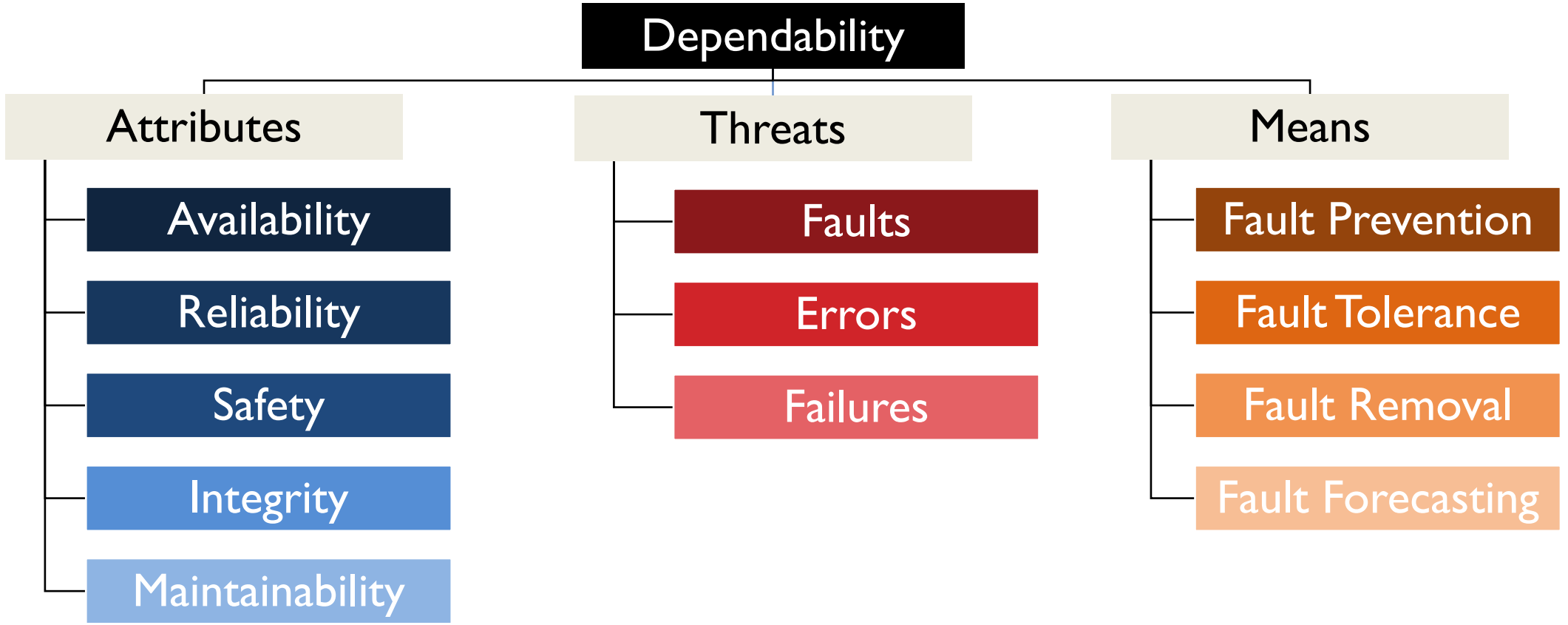
There are several opportunities for reliability engineers in the gap separating the raw technology from the mission of the final system

Do the right thing for reliability/resiliency at the "best" places in the system stack

Optimize reliability solutions across the entire system stack to provide the most optimal & cost-effective implementations

DEPENDABILITY

Specific concepts



RELIABILITY METRICS

How to measure the reliability of a system?

Failure rate (λ): Number of failures per unit of time. If the failure rate λ is constant, reliability can be modeled using an exponential distribution: $R(t) = e^{-\lambda t}$

Failure in time (FIT): Number of failures in 10^9 device hours.

Mean Time To Failure (MTTF): Arithmetic mean (average) time to failure of a system. Usually expressed in hours.

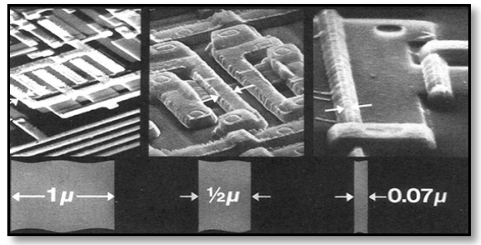
$MTTF = \frac{1}{\lambda}$

Executions per Failures (EPF): EPF is the number of times an application must be executed before observing a system failure. It is computed as $EPF = \frac{EIT}{\lambda}$ where EIT (Executions in Time) is the number of executions of an application in 10^9 hours of device operation. The EPF enables to jointly analyze performance and reliability into a single metric.

MANY MORE …

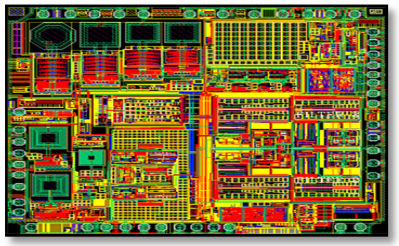
THE RELIABILITY TAX

Techniques to handle reliability are already there



Process: Transistor architecture/geometry, doping details, FinFET fin height, buried gate

Circuit: Radiation resistant circuits, Razor latches, Tunable Replica Circuits, LEAP-DICE

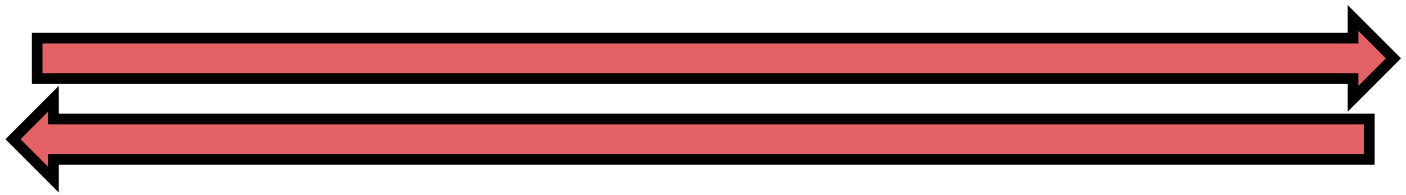


Architecture/Microarchitecture solutions: Parity, ECC, TMR, Lockstep, Watchdogs

Software solutions:

EDDI, CFCSS, ED4I, ABFT, RECCO, CFRE

Techniques that can address multiple reliability issues with a single mechanism are most cost-effective (e.g., ECC).



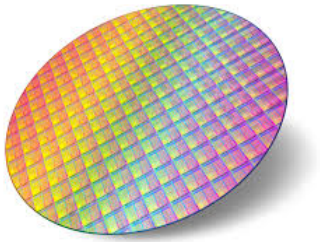
No clear answer

Best rule?

THE RELIABILITY TAX

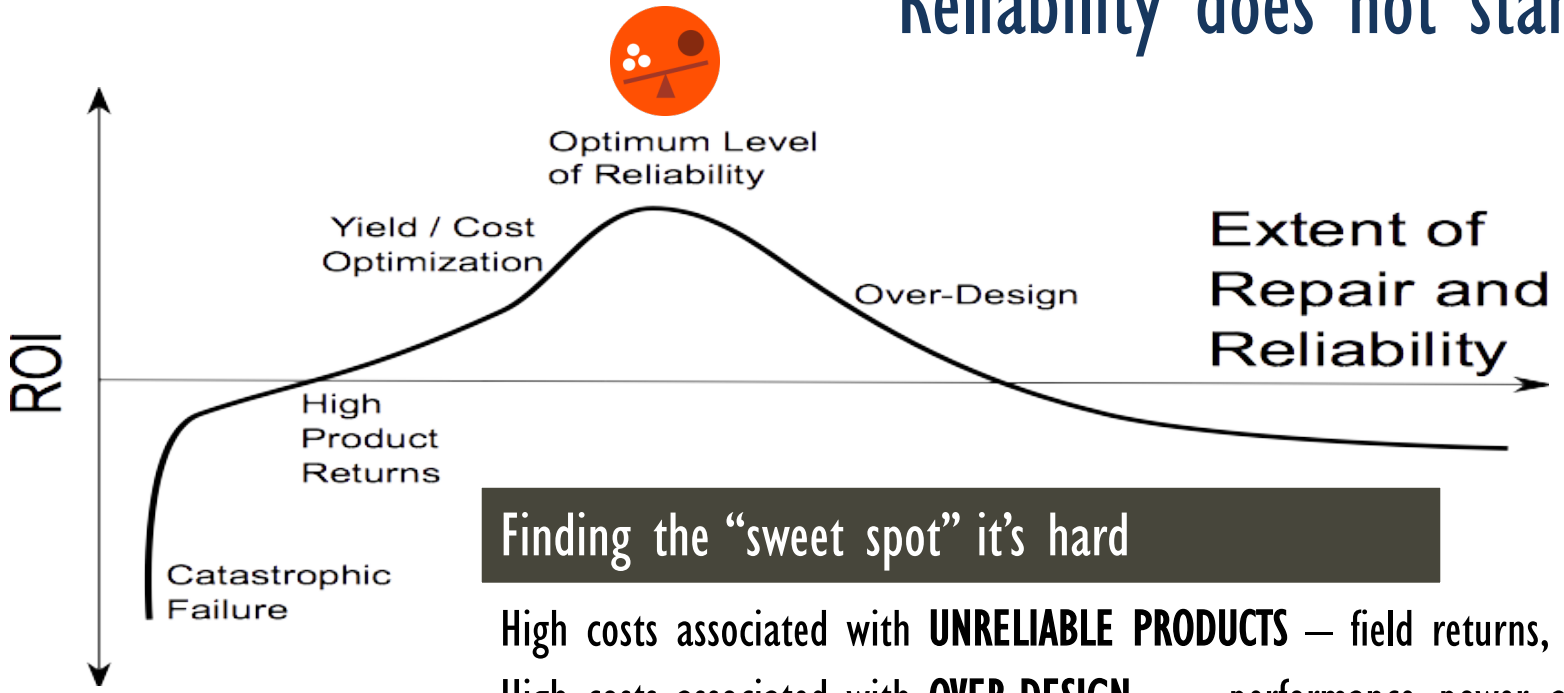
Reliability does not stand alone

Performance



Area

Power



Finding the "sweet spot" it's hard

High costs associated with **UNRELIABLE PRODUCTS** – field returns, reputation

High costs associated with **OVER-DESIGN** – performance, power, area

How can we help alleviating the reliability "tax"?

- Do we really need to protect everything?
- Do we really need to protect everything all the time?
- Can reliability mechanisms be re-purposed when not needed?

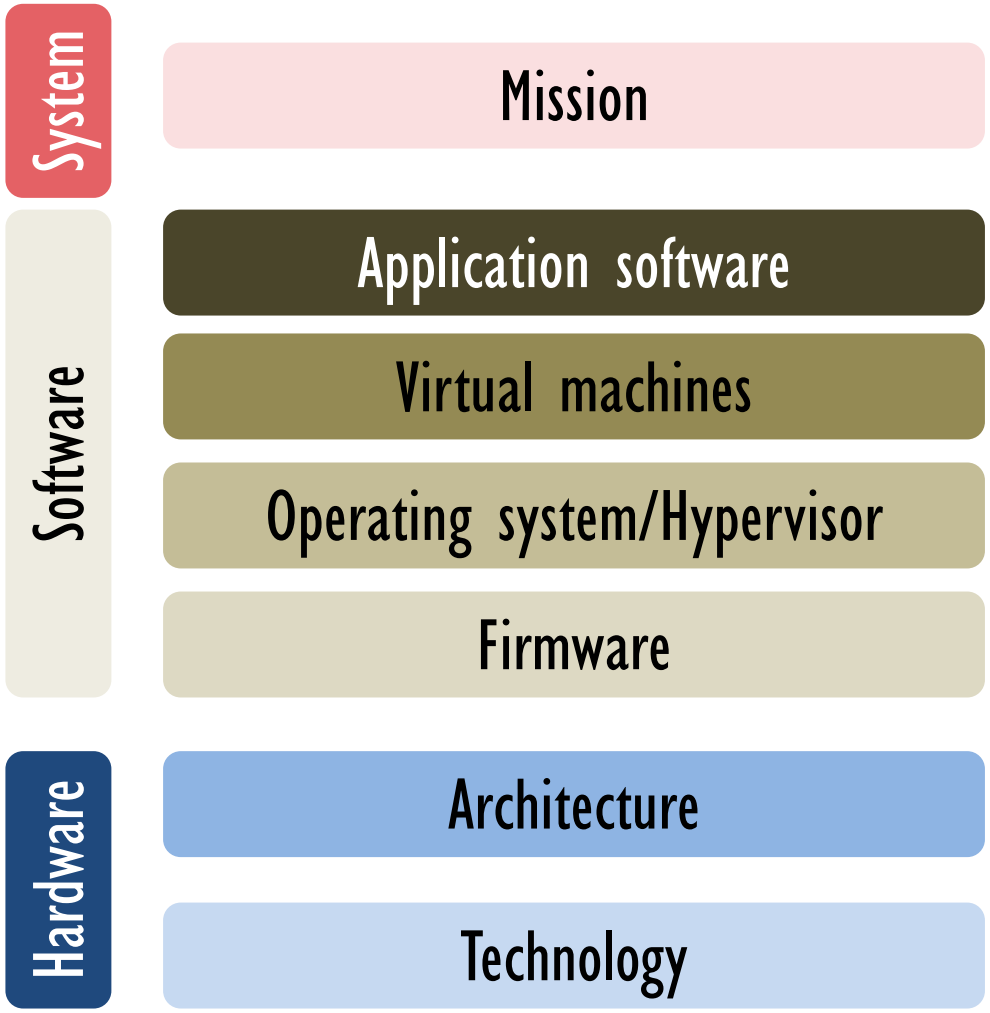
CROSS LAYER RESILIENCE

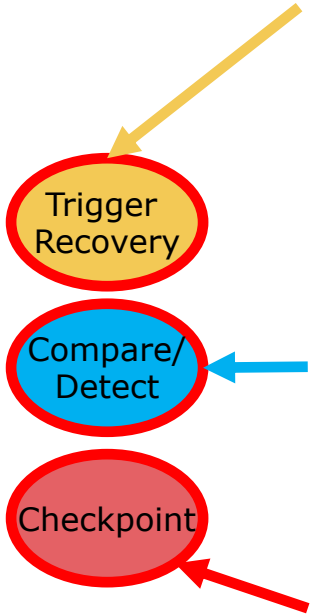
The Path to Optimal Reliability?

- Promises, promises…
- Lots of work out there on cross-layer resilience
- Still limited impact on the market

CROSS LAYER RESILIENCE

Faults can be dealt with at different levels of the stack





Recovery is triggered at the OS level

- OS knows what is affected by the data
- OS can roll back the appropriate operations
- OS can manage the checkpoints
- Recovery decisions can be made with longer latencies

Firmware can manage the error signals

- Compare checkpoints if needed
- Detect/maintain the last known good checkpoint

Hardware takes periodic checkpoints

- Lowest latency and best granularity

CROSS LAYER RESILIENCE

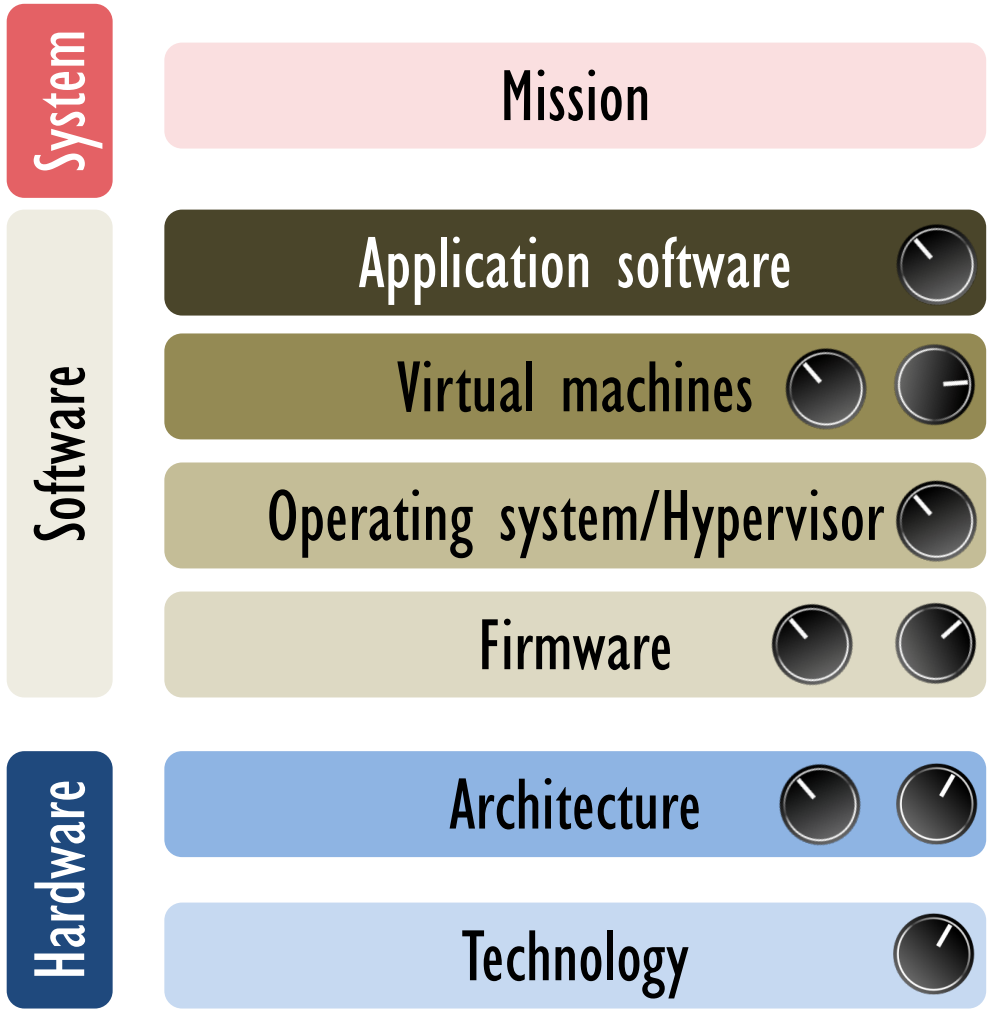
The ultimate solution to the reliability problem

- Can address multiple fault types
- Can minimize/eliminate the "tax" by amortizing across the system stack
- Can take system level decisions incurring into penalties when a fault is detected and will actually impact the mission in a meaningful way

Great, right? SO, what's the problem?

CROSS LAYER RESILIENCE

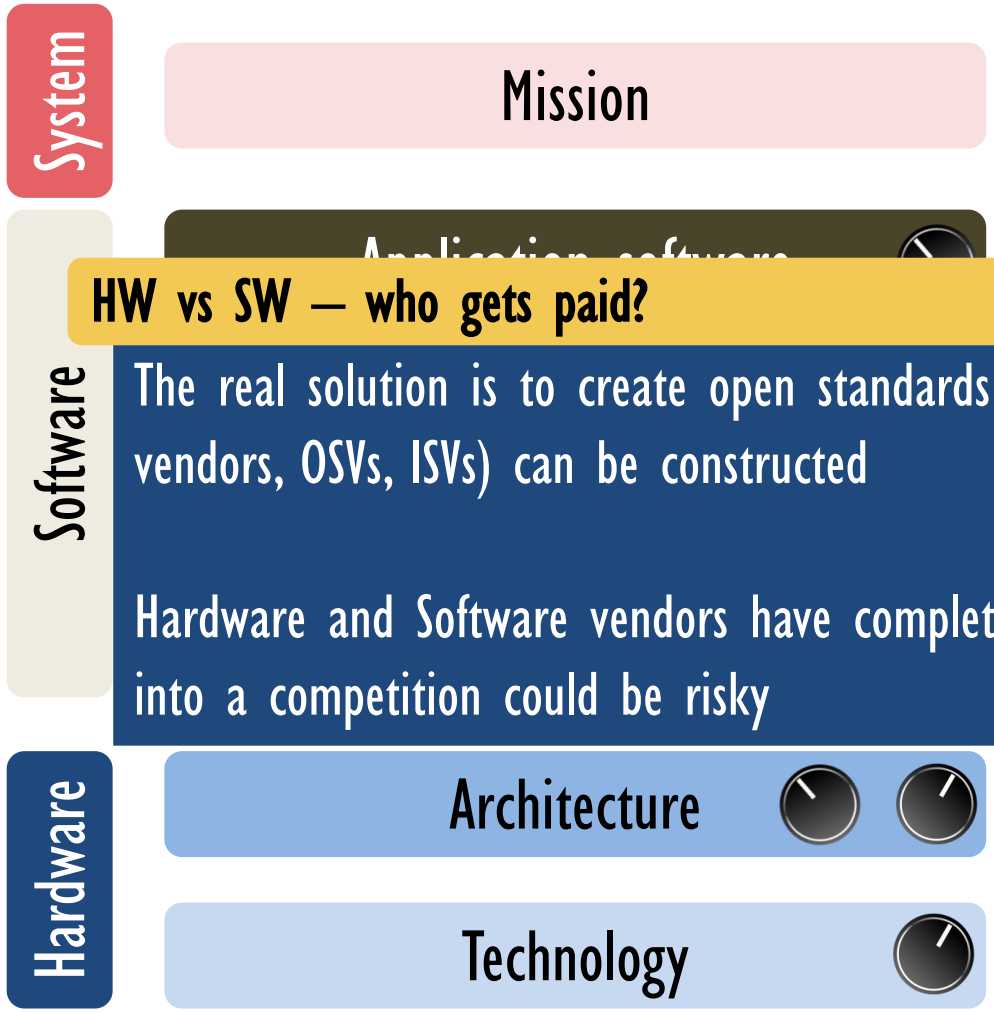
Key problem: assumes the vendor owns or influences the entire system stack



- Very slow/minimal penetration into industry unless a single entity provides a full stack solution
- It requires HW/SW vendors to introduce reliability solutions that can be tuned or fully disabled for markets where not needed
- It should look at schemes providing savings in complexity/power/performance/area, especially in cases where coverage can be deemed sufficient
- Requires HW & SW to “speak” to each other

CROSS LAYER RESILIENCE

Key problem: assumes the vendor owns or influences the entire system stack



- Very slow/minimal penetration into industry unless a single entity provides a full stack solution

HW vs SW – who gets paid?

The real solution is to create open standards around which the entire ecosystem (HW vendors, OSVs, ISVs) can be constructed

Hardware and Software vendors have completely different business models, pushing them into a competition could be risky

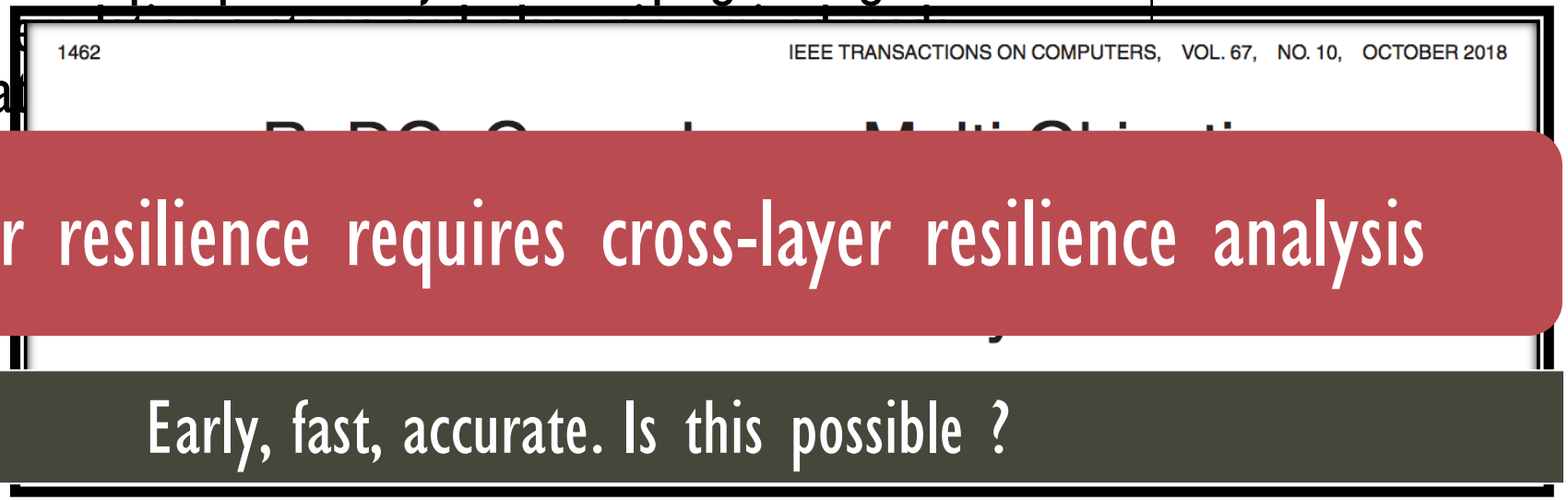
- Requires HW & SW to "speak" to each other

CROSS LAYER RESILIENCE

Key Problem: too many design alternatives, EDA tools not mature

CLEAR: Cross-Layer Exploration for Architecting Resilience:

- Two processors: SPARC Leon3, Alpha IVM
- 18 benchmarks: SPECINT2000, DARPA PERFECT
- 10 error correction techniques at different levels of the stack
- 9 million flip-flop error injection campaign using three BEE3 FPGA
  simula



Cross-layer resilience requires cross-layer resilience analysis

Early, fast, accurate. Is this possible ?

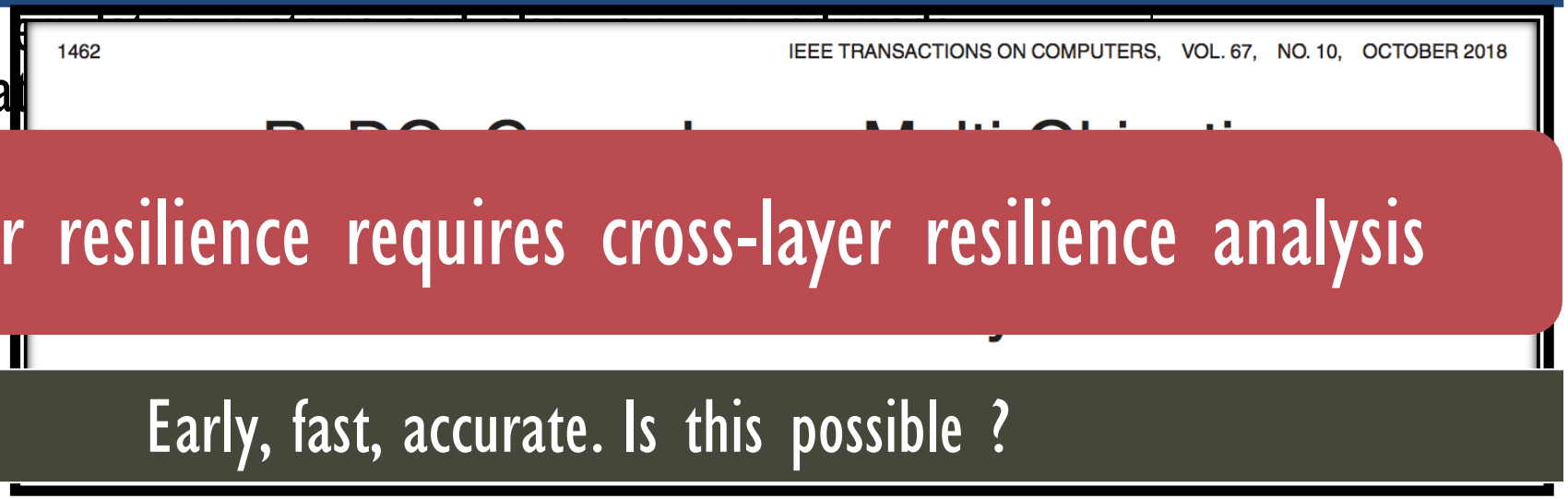
CROSS LAYER RESILIENCE

Key Problem: too many design alternatives, EDA tools not mature

CLEAR: Cross-Layer Exploration for Architecting Resilience:

- **Can results of this analysis be extended to other cases?**
- The general answer is no.
- Every application is different and to really get benefits from the application of cross-layer resilience techniques it must be analyzed and optimized
- But system designers cannot afford to repeat the same analysis for every design

FPGA
simula

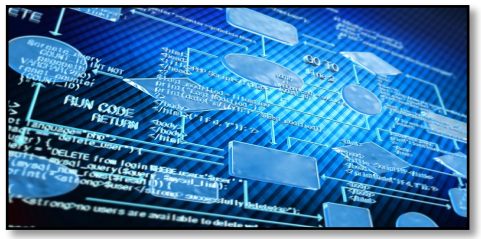


Cross-layer resilience requires cross-layer resilience analysis

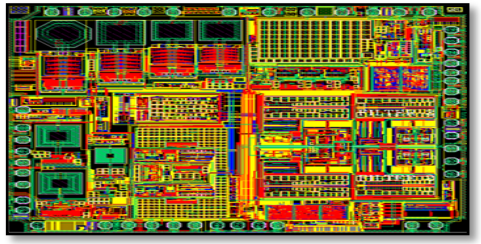
Early, fast, accurate. Is this possible ?

CROSS LAYER RELIABILITY

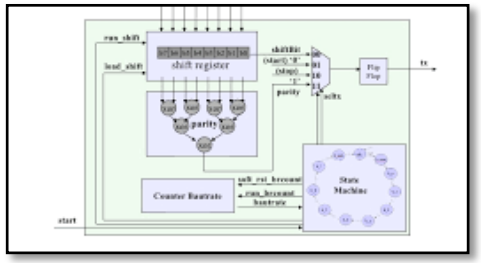
EDA tools for cross layer reliability analysis



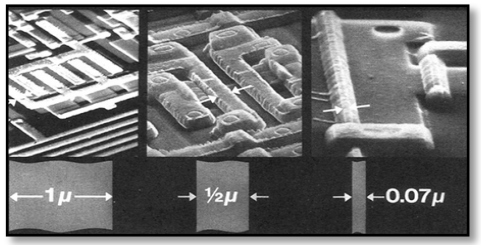
software



uArch



RTL



Circuit

speed

accuracy

Speed

Accuracy

Trade-off

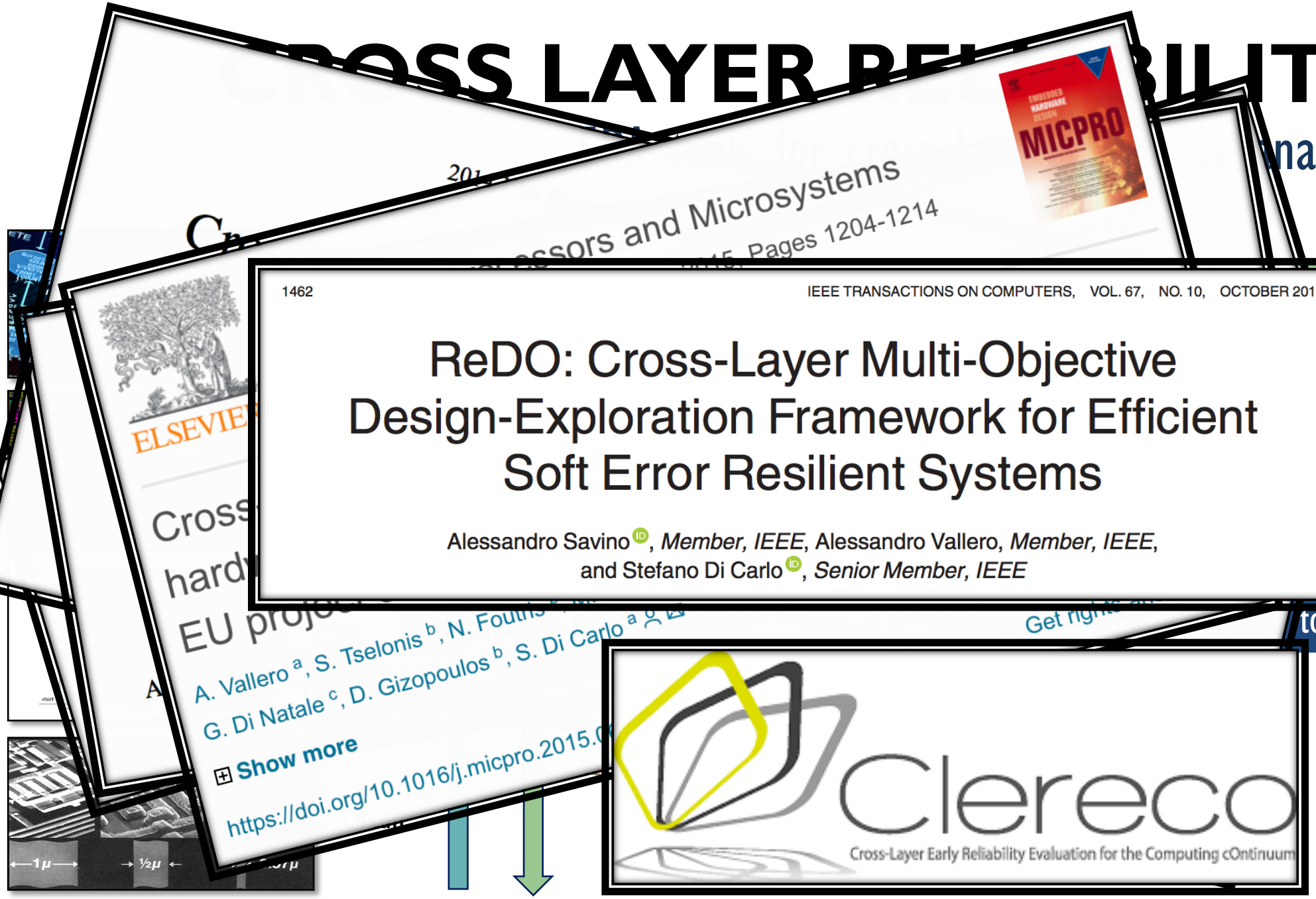
Any preferred choice?

No single approach can meet the requirements.

Stochastic models combining contributions are the key to succeed

CROSS LAYER RELIABILITY

analysis



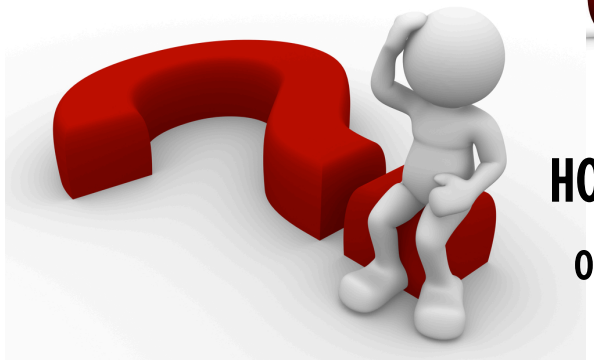
to succeed

CONCLUSIONS

Cross-Layer Soft-Error Resilience of Computing Systems

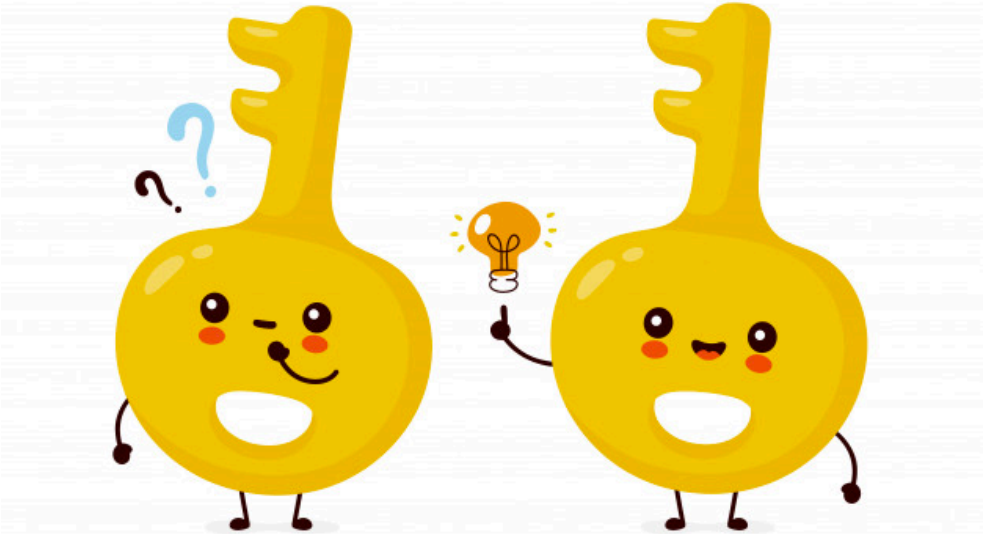
WHAT is it?

WHY do we need it?



HOW do we obtain it?

CROSS-LAYER RESILIENCE ANALYSIS



Want to know more?
Keep watching the tutorial.

Questions?

http://www.testgroup.polito.it