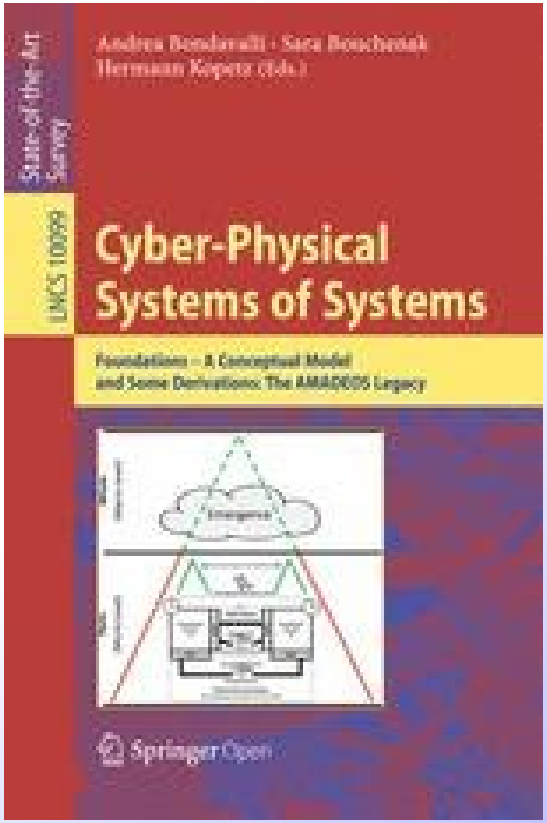
Basics on Cyber-Physical SoSs

Andrea Ceccarelli
Dipartimento di Matematica e Informatica,
Universita' di Firenze

andrea.ceccarelli@unifi.it

Teaching Material

- **Cyber-Physical Systems of Systems**
  Foundations – A Conceptual Model and Some Derivations: The AMADEOS Legacy
- Editors: Andrea Bondavalli, Sara Bouchenak, Hermann Kopetz
  ISBN: 978-3-319-47589-9 (Print) 978-3-319-47590-5 (Online)



➢ Chapter 1 – Basic Concepts on Systems of Systems

The SoSs Era

Computing evolution

- Mainframe computing (60's-70's)
  - Large computers to execute big data processing applications
- Desktop computing & Internet (80's-90's)
  - One computer at every desk to do business/personal activities
- Ubiquitous computing (00's)
  - Numerous computing devices in every place/person
  - "Invisible" part of the environment
  - Millions for desktops vs. **billions** for embedded processors
- Cyber Physical Systems (10's)

What are Cyber-Physical Systems?

- Cyber – computation, communication, and control that are discrete, logical, and switched
- Physical – natural and human-made systems **governed by the laws of physics** and operating in continuous time
- Cyber-Physical Systems – a system consisting of a computer system (the cyber system), a controlled object (a physical system) and possibly of interacting humans.

"CPS will transform how we interact with the physical world just like the Internet transformed how we interact with one another." [Fei Hu. Cyber-Physical Systems. CRC press. 2013]

From Monolithic Systems …

Starting from **MainFrames**, computers were usually characterized by distinguishable services that are not clearly separated in the implementation but are interwoven,

- for example
  - data input and output,
  - data processing,
  - error handling, and
  - the user interface,
- rather than containing separate components

- Such "monolithic" architecture defines the **monolithic software systems**

Monolithic: term still in use?

Monolithic kernel: all services (file system, VFS, device drivers, etc) as well as core functionality (scheduling, memory allocation, etc.) are a tight knit group **sharing the same space**

Microkernel: core functionality is isolated from system services and device drivers (which are basically just system services). For instance, VFS (virtual file system) and block device file systems (i.e. minixfs) are separate processes that run outside of the kernel's space, using IPC to communicate with the kernel, other services and user processes.

- **Linux is a monolithic kernel (but it is modular!!!)**
  - if it's a module in Linux, it's a service in a microkernel, indicating an isolated process.

… to Systems of Systems (SoSs)

However, many of the established assumptions in classical system design, such as

- the scope of the system is known,
- the design phase of a system is terminated by an acceptance test,
- faults are exceptional events,

are not always justified in modern cyber-physical systems.

Are the above assumptions valid?

Helicopter

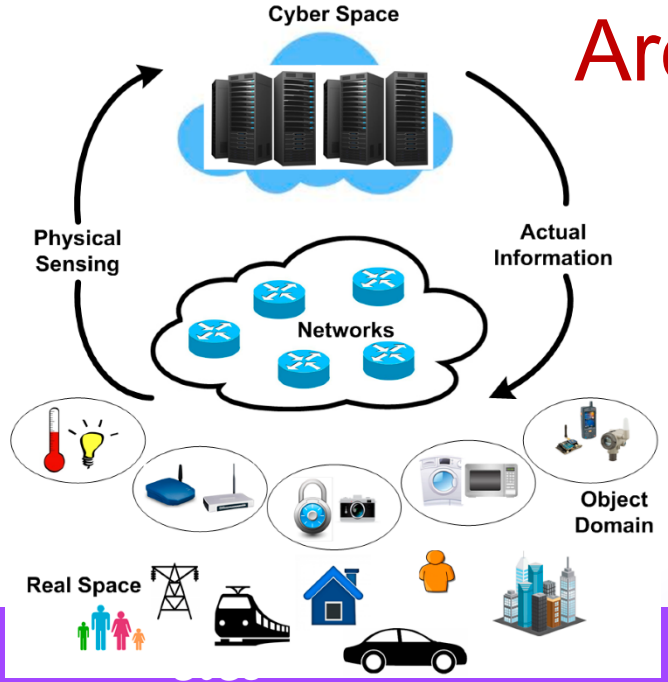
Railway infrastructure

Smart grids

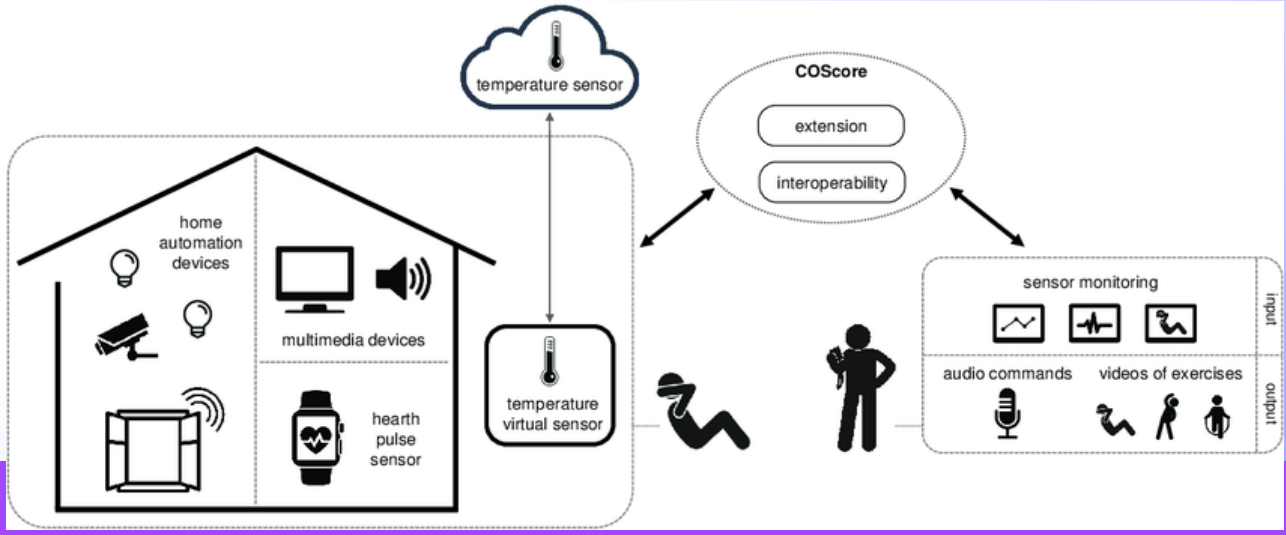
Vehicles in a traffic jam

What happens in SoSs

A **System of System (SoS)** stems from the integration of existing systems (legacy systems), normally operated by different organizations, and new systems that have been designed to take advantage of this integration.

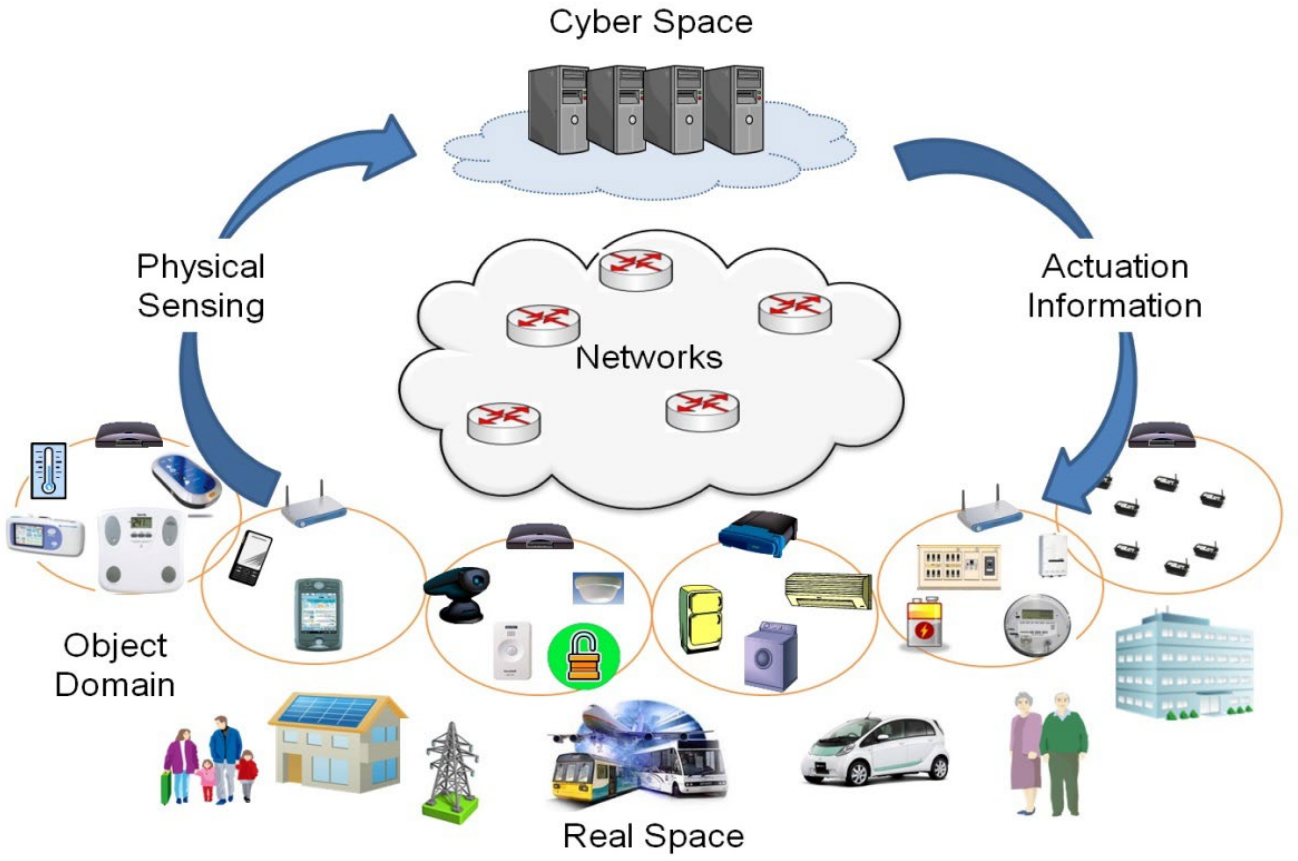


Are the previous assumptions valid?

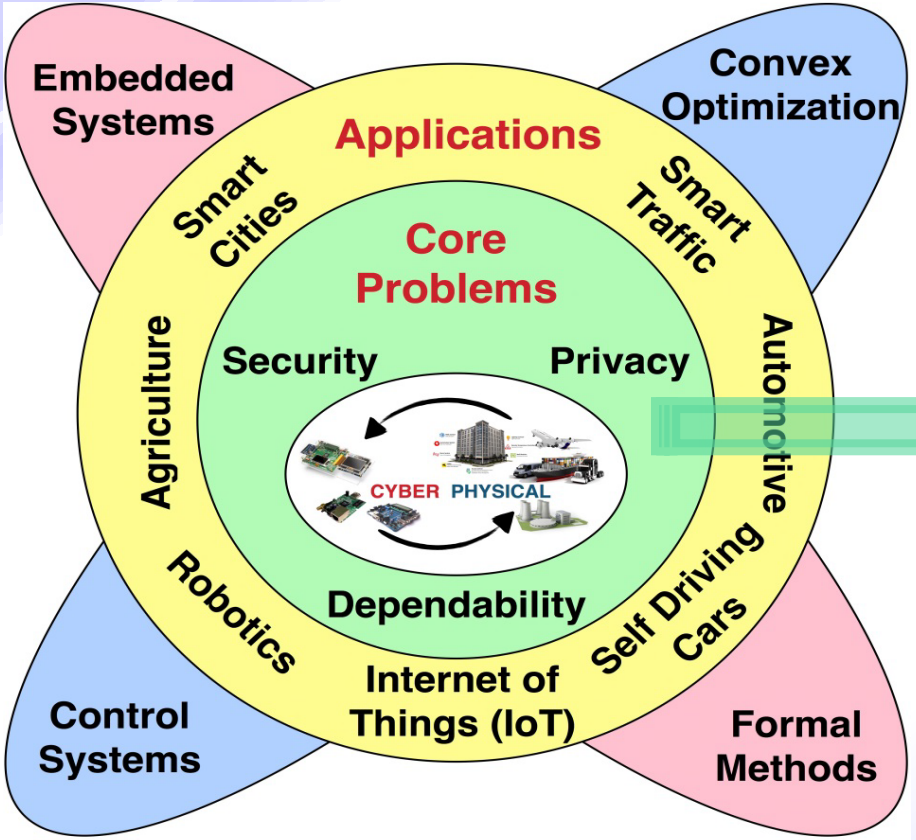


CPS view

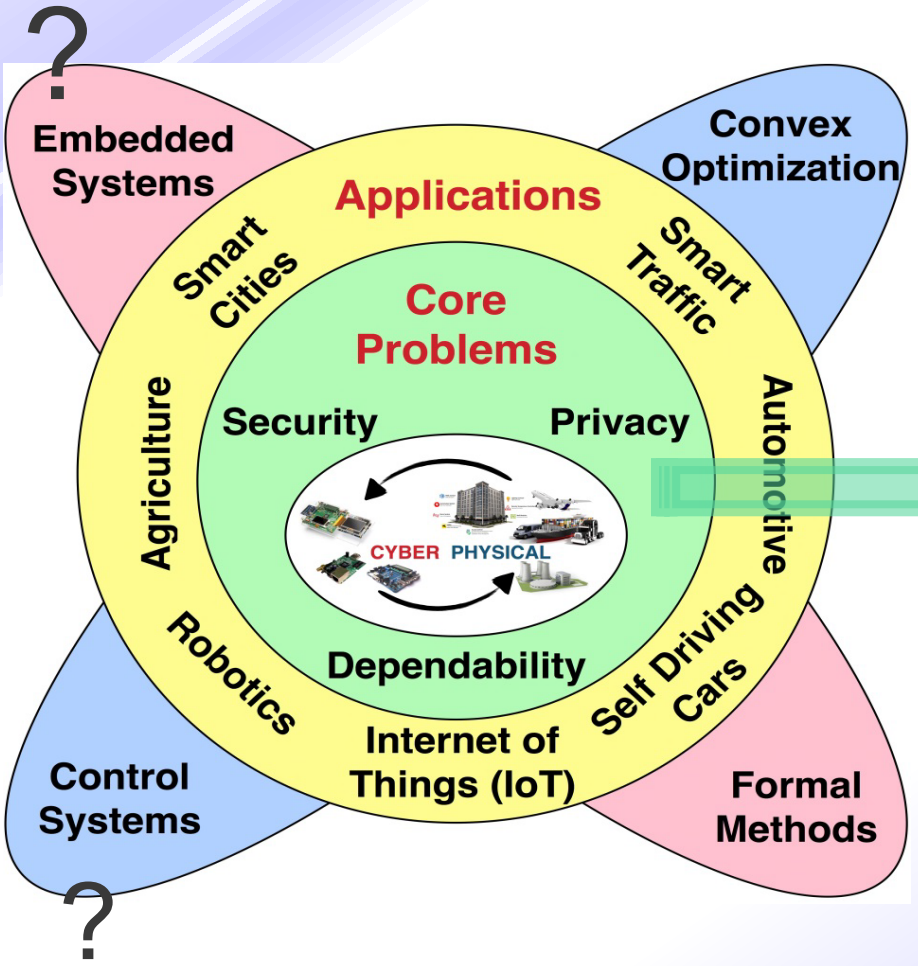
CPSs are physical and engineered systems whose operations are monitored, coordinated, controlled and integrated by a computing and communication core.



Core problem: how to make such systems Secure, Resilient, ...

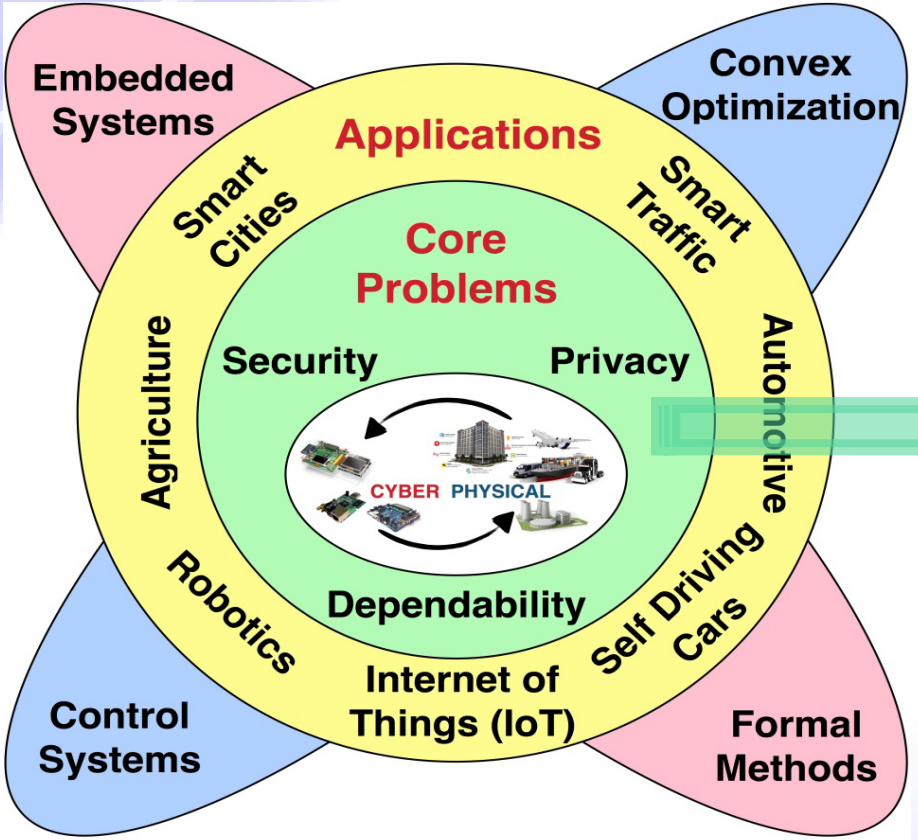


Core problem: how to make such systems Secure, Resilient, ...



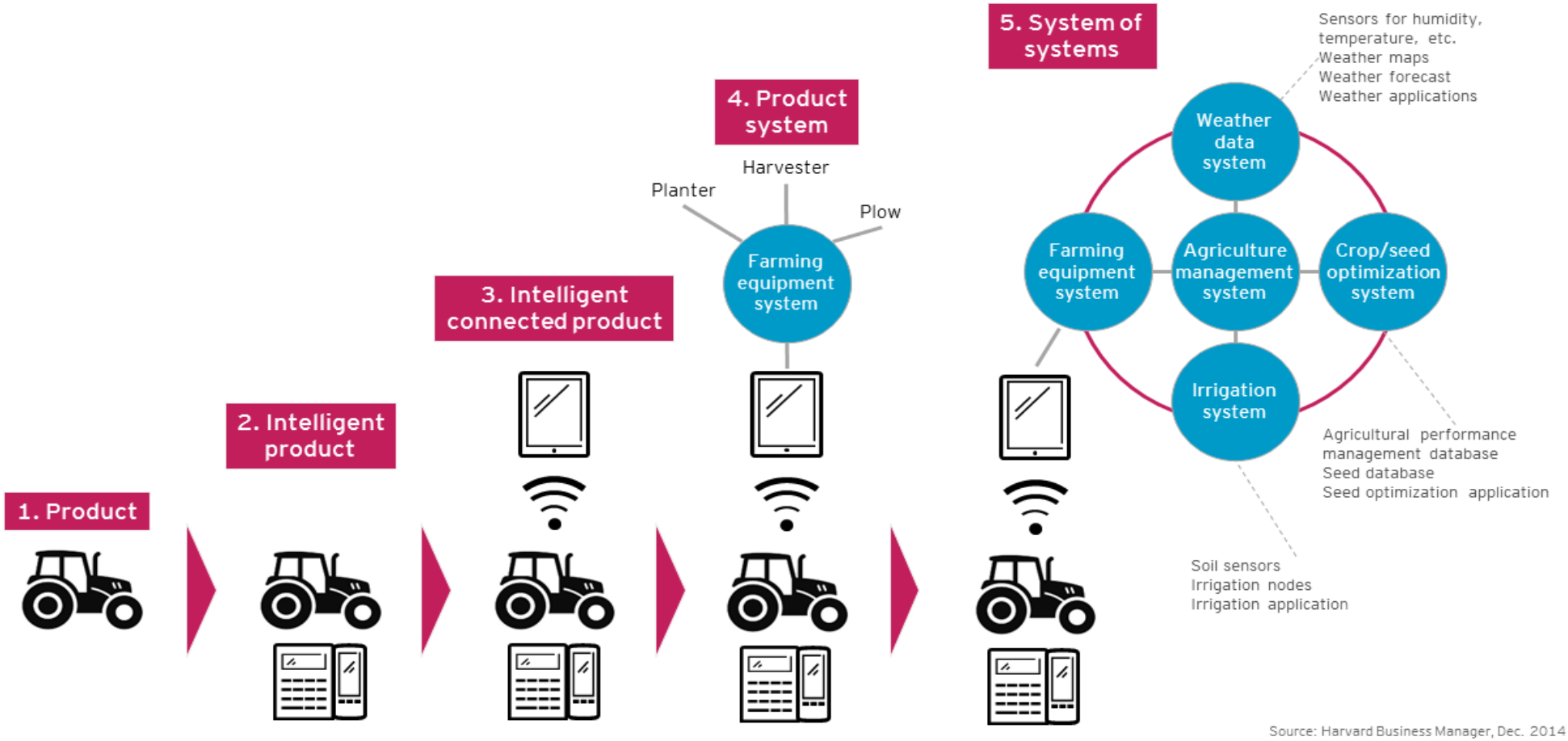
Core problem: how to make such systems Secure, Resilient, ...

computer system that has a dedicated function within a larger mechanical or electronic system



control system manages, commands, directs, or regulates the behavior of other devices or systems using control loops.

CPS... and related terms: (Cyber-Physical) Systems of Systems



CP-SoS?

Updates from 2022 HiPEAC workshops on CP-SoS→

SoS is now viewed as an essential part of CPS – so the acronym CPSoS is being phased out to support such an evolution […].

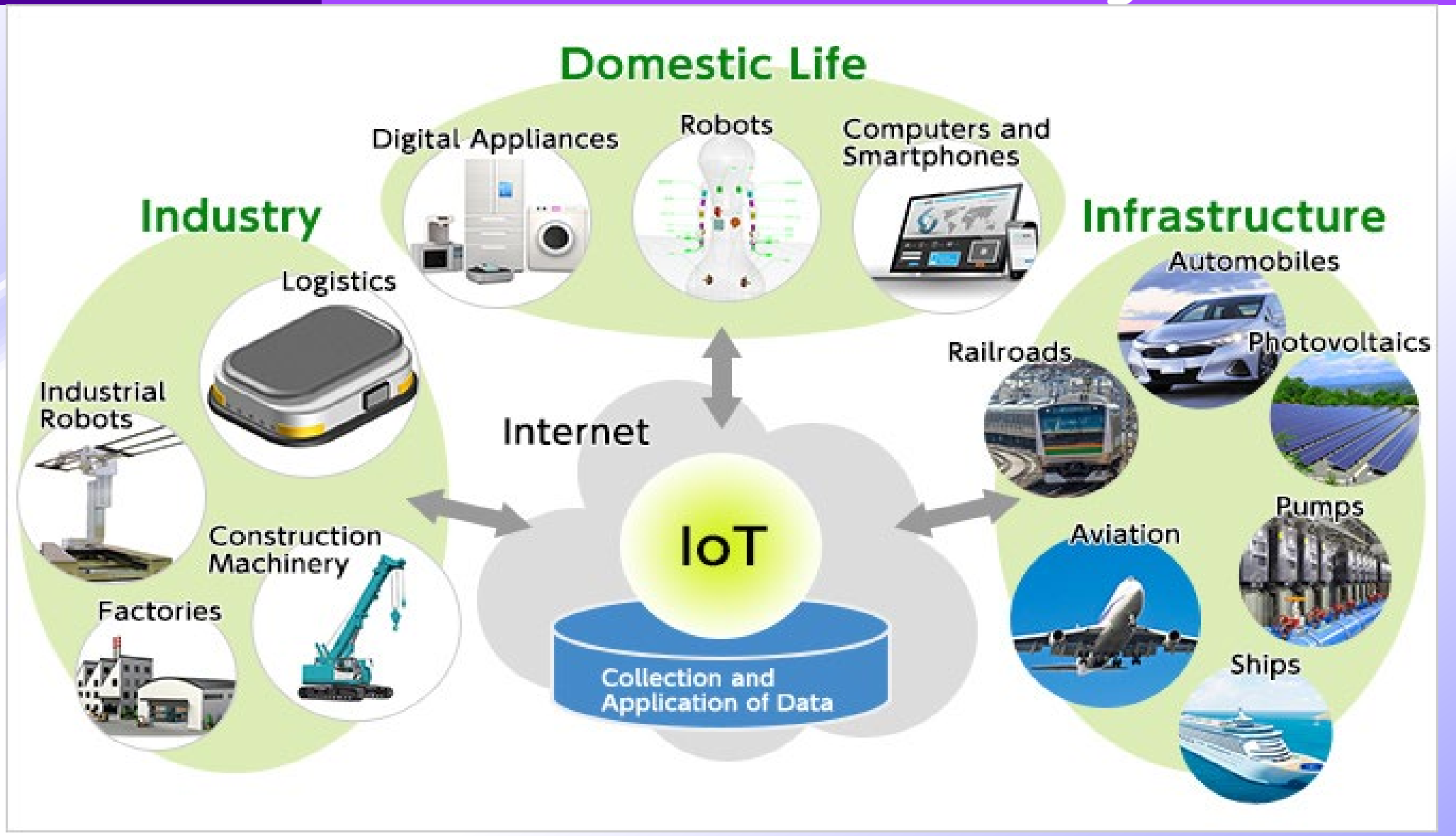
For instance, the acronym is no longer present in the challenges if you compare the latest two versions of the ECS SRIA 2023

https://ecssria.eu/ECS-SRIA%202023.pdf → *From Page 135, chapter 1.4 Foundational Technology Layer - System of Systems*

The System of Systems (SoS) technology layer represents the upper layer of ECS technology stack for digitalisation solutions. This technology layer emerges from the composition of embedded and cyber-physical systems (CPS), connectivity and distributed software platforms.

Electronic Components and Systems (ECS) SRIA 2023: electronic components and systems – Strategic research and innovation agenda 2023

CPS... and related terms: *Internet of Things*



More on the Internet of Things (IoT)

The Internet of Things enables objects **sharing information with other objects**/members in the network, **recognizing events** and changes so to **react autonomously** in an appropriate manner.

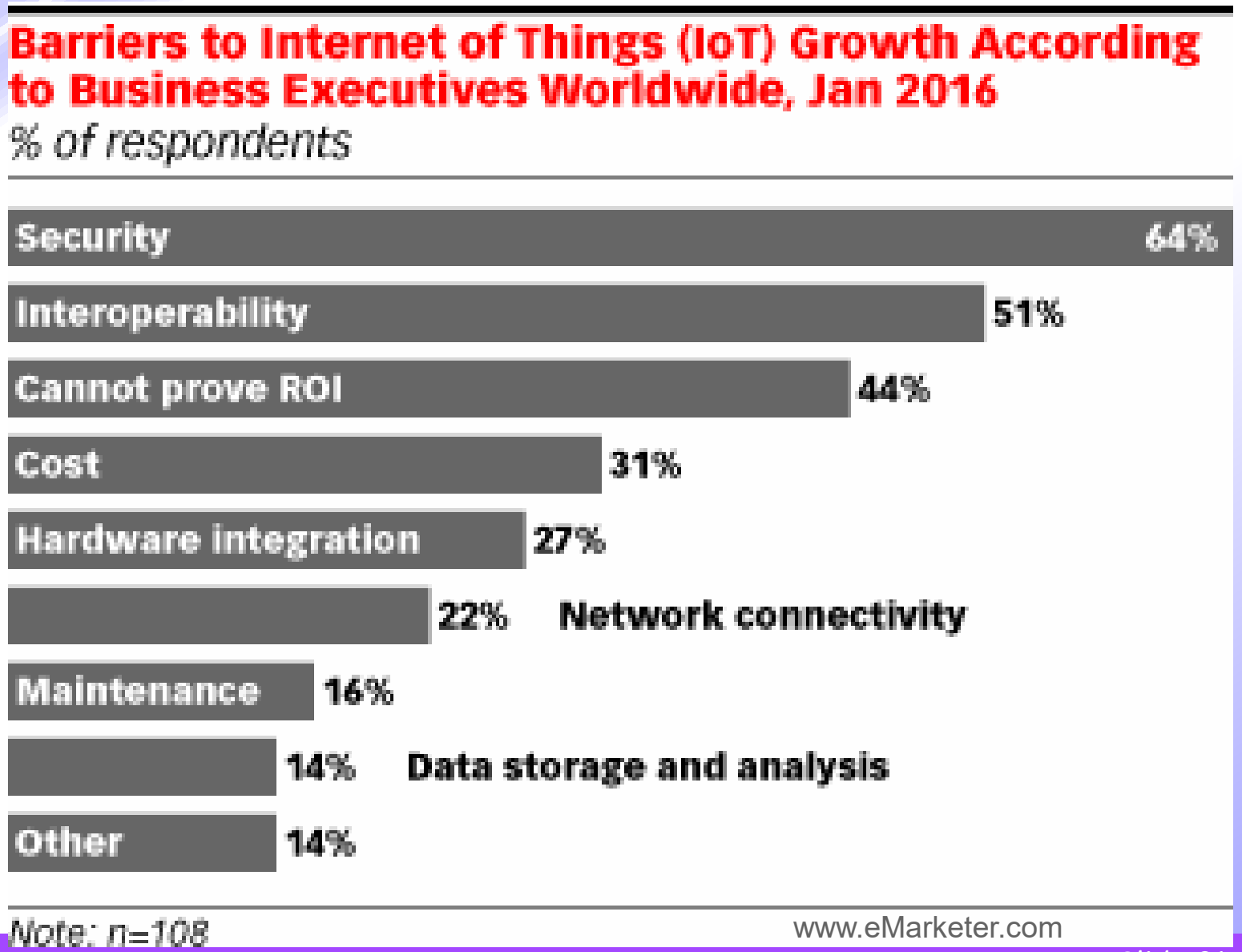
The IoT therefore builds on communication between things (machines, buildings, cars, animals, etc.) that **leads to action and value creation.**

- European Commission, 2014

Khan, R., et al. Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. FIT 2012.

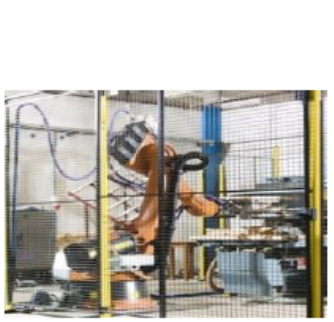
Let's reason again on our previous assumptions

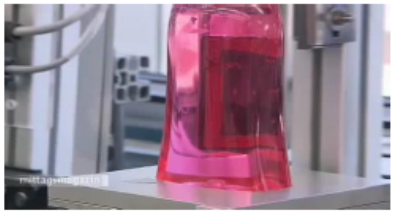
Do our previous assumptions fit IoT?

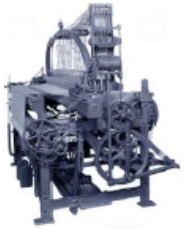


CPS & Industry 4.0

From Industry 1.0 to Industry 4.0: Towards the 4th Industrial Revolution







010001101
001010100
100101010
010010101

4. Industrial Revolution
based on Cyber-Physical Production Systems
Industrie 4.0

3. Industrial Revolution
through Introduction of electronics and IT for a further automization of production
Industrie 3.0

2. Industrial Revolution
through introduction of mass production based on the division of labour powerde by electrical energy
Industrie 2.0

First **Mechanical Loom** 1784

1. Industrial Revolution
through introduction of mechanical production facilities powered by water and steam
Industrie 1.0

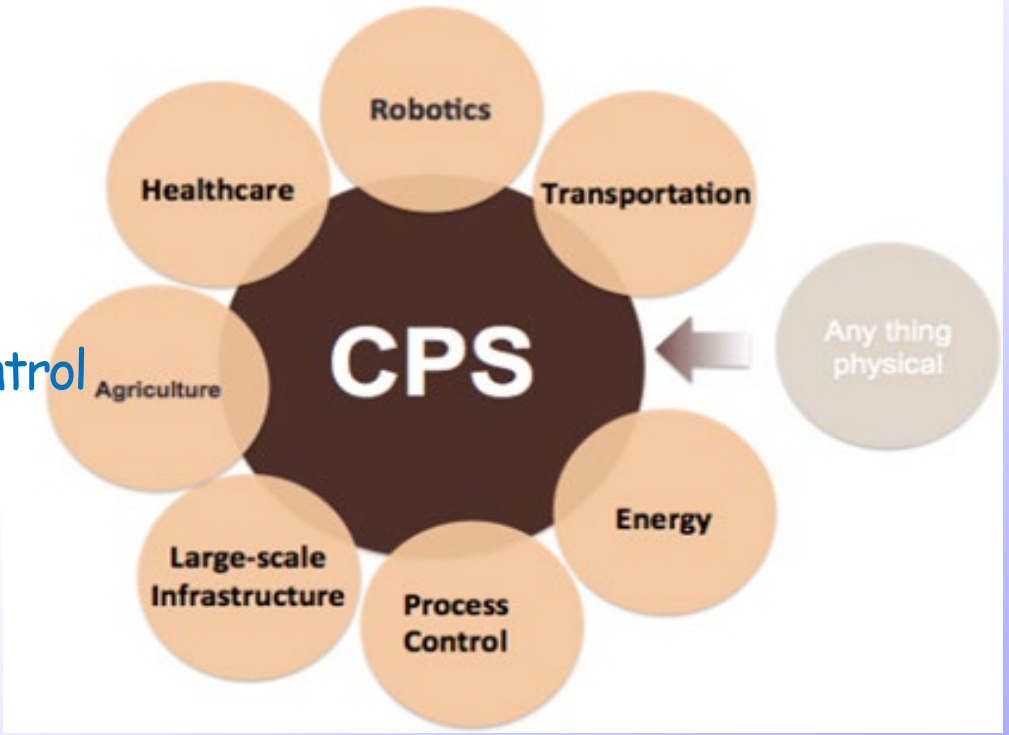
Degree of Complexity

End of 18th Century | Start of 20th Century | Start of 70ies | today | t

Application Domains of Cyber-Physical Systems

- Healthcare
  - Medical devices
  - Health management networks
- Transportation
  - Automotive electronics
  - Vehicular networks and smart highways
  - Aviation and airspace management
  - Avionics
  - Railroad systems
- Process control
- Large-scale Infrastructure
  - Physical infrastructure monitoring and control
  - Electricity generation and distribution
  - Building and environmental controls
- Defense systems
- Tele-physical operations
  - Telemedicine
  - Tele-manipulation

Let's pick some examples. Can you discuss what is the CPS, and what is the SoS? Can you discuss criticalities?



Multi-Level Hierarchy

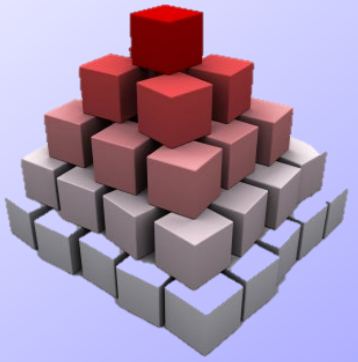
SoSs are characterized by a **multi-level** hierarchy, or rather a recursive structure where:

- a system, the whole at the level of interest (the macro-level), can be taken apart into
- a set of subsystems, the parts, that interact statically or dynamically at the level below (the micro-level).

Each one of these subsystems can be viewed as a system of their own.

Recursion ends when the internals of a subsystem is **of no further interest**.

- we call such a subsystem at the lowest level of interest - the base of the hierarchy - an elementary part or a **component**



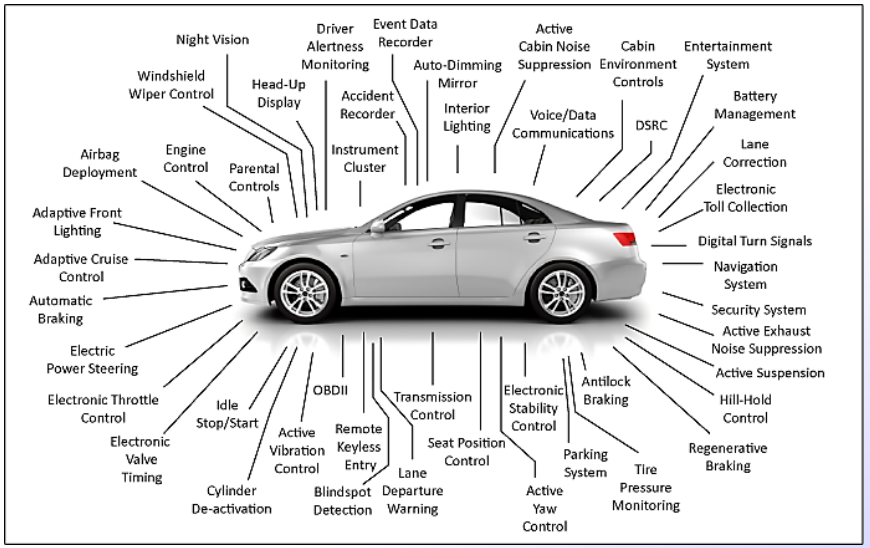
Examples of multi-level hierarchy

- Can you propose an example of multi-level hierarchy?
  - a system, the whole at the level of interest (the macro-level), can be taken apart into
  - a set of subsystems, the parts, that interact statically or dynamically at the level below (the micro-level).

www.wooclap.com

→ QDMTNK

One example of multi-level hierarchy



Main Differences

The main differences between the two approaches can be summarized as follows

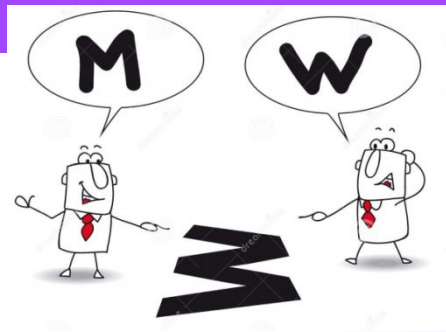
Characteristic	Monolithic	System-of-system
Scope of the System	Fixed (Known)	Unknown
Clock Synchronization	Internal	External e.g., GPS
Structure	Hierarchical	Networked
Requirements and Spec.	Fixed	Changing
Evolution	Version Control	Uncoordinated
Testing	Test Phases	Continuous
Implementation	Technology Given and Fixed	Unknown
Faults (Physical, Design)	Exceptional	Normal
Control	Central	Autonomous
Emergence	Insignificant	Important
System Development	Process Model	???

Viewpoints (I)

To reduce the *cognitive effort* needed to comprehend the behaviour of an SoS, its main characteristics can be summarized as viewpoints, or rather simplified dimensions of analysis:

- Fundamental System Concepts
  - The definition of an SoS and its related parts
- Time
  - The progression of time and its role in an SoS
- Data and state
  - the data and information exchanged between the parts of an SoS
- Actions and Behaviour
  - the dynamics of an SoS, either event-based view or a state-based view

Viewpoints (II)



- Communications
  - the role of a communication system in an SoS
- Interfaces
  - the interaction of components with each other and with the environment
- Evolution and Dynamicity
  - SoS dynamicity, intended as short term changes, and evolution (long term changes)
- System design and tool
  - The concepts to define design methodologies to engineer SoSs
- Dependability and Security (plus privacy)
  - Dependability and security concepts, in compliance with existing taxonomies
- **Emergence**

Fundamental System Concepts

Fundamentals

Domain: The Domain comprises the set of entities and the relations among the entities that are of interest when modeling the selected view of the world.

To structure the domain, we must identify objects that have a distinct and self-contained existence.

- **Entity**: Something that exists as a distinct and self-contained unit.
  - **Thing**: A physical entity that has an identifiable existence in the physical world.
  - **Construct**: A non-physical entity, a product of the human mind, **such as an idea.**

Let's build our example

By now, you should be able to mention a reference domain…

- Decide on a domain.
- What are their entities (things + construct)?
- And what are their relations?

(we will choose one as example)

www.wooclap.com →
QDMTNK

System

System: An entity that is capable of **interacting with its environment** and may be **sensitive to the progression of time.**

Note that the system may react differently, to the same pattern of input activity, depending on the environment e.g., a time-controlled heating system.

Environment of a System: The entities and their **actions** in the domain that are not part of a system but have the capability to interact with the system.

- System and Environment are separated by a **System Boundary**, a dividing line between two systems or between a system and its environment.

System Architecture

System Architecture: The blueprint of a design that establishes the overall structure, the major building blocks and the interactions among these major building blocks and the environment.

When designing the system, every organization that develops a system follows a set of explicit or implicit rules and conventions, such as

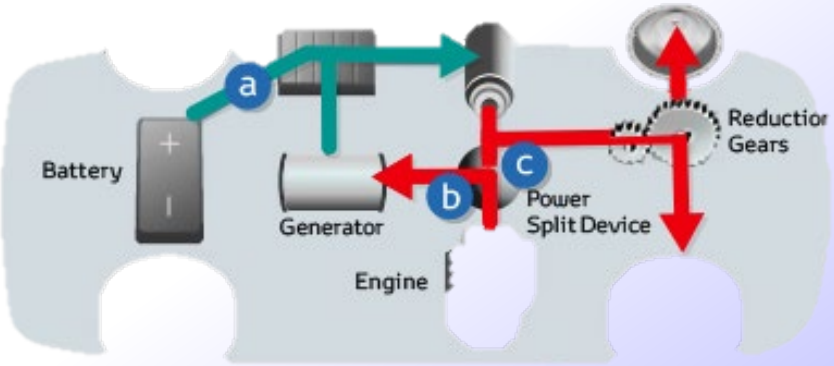
- naming conventions,
- representation of data (e.g., endianness of data),
- protocols

These explicit or implicit rules and conventions are called the **architectural style**.

Boundaries in SoSs

In SoS Engineering, such an approach can be problematic, because in many SoS the system boundary may change frequently.

- Consider a car-to-car SoS that consists of a plurality of cars cruising in an area.

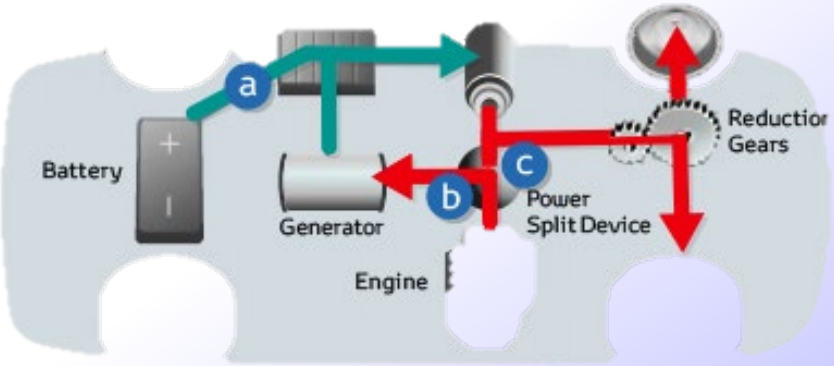


Where is the boundary of such an SoS?

Boundaries in SoSs

In SoS Engineering, such an approach can be problematic, because in many SoS the system boundary may change frequently.

- Consider a car-to-car SoS that consists of a plurality of cars cruising in an area.



Where is the boundary of such an SoS?

it is hardly possible to define a stable boundary of an SoS

Autonomous System

- In the above example of a car-to-car SoS each individual car in the system
  - consisting of the mechanics of the car, the control system, and the driver
- Can be considered as an autonomous system that tries to achieve its given objective without any control by another system.

Autonomous System: A system that can provide its services without guidance by another system.

Cyber-Physical System

Many systems are composed of (autonomous) interrelated parts, each of them hierarchic in structure until some lowest level of elementary subsystem, a subordinate system that is a part of an encompassing system.

Constituent System (CS): An autonomous subsystem of an SoS, consisting of computer systems and possibly of controlled objects and/or human role players that interact to provide a given service.

Cyber-Physical System (CPS): A system consisting of a computer system (the cyber system), a controlled object (a physical system) and possibly of interacting humans

Systems of Systems

System-of-Systems (SoS):

An SoS is an integration of a finite number of constituent systems (CS) which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal (*Jamshidi*).

Note: boundaries are defined for a period of time, then they may change

Note: "finite" helps in defining the limits of the boundaries: can you think of/elaborate an example where assumptions on boundaries are necessary?

Types of Systems of Systems

Directed SoS: An SoS with a central managed purpose and central ownership of all CSs.

- An example?

Acknowledged SoS: Independent ownership of the CSs, but cooperative agreements among the owners to an aligned purpose.

- An example?

Collaborative SoS: Voluntary interactions of independent CSs to achieve a goal that is beneficial to the individual CS.

- An example?

Virtual SoS: Lack of central purpose and central alignment.

- An example?

Types of Systems of Systems

Directed SoS: An SoS with a central managed purpose and central ownership of all CSs.

- The set of control systems in an unmanned rocket.

Acknowledged SoS: Independent ownership of the CSs, but cooperative agreements among the owners to an aligned purpose.

- railway

Collaborative SoS: Voluntary interactions of independent CSs to achieve a goal that is beneficial to the individual CS.

- ATM, bitcoins miners

Virtual SoS: Lack of central purpose and central alignment.

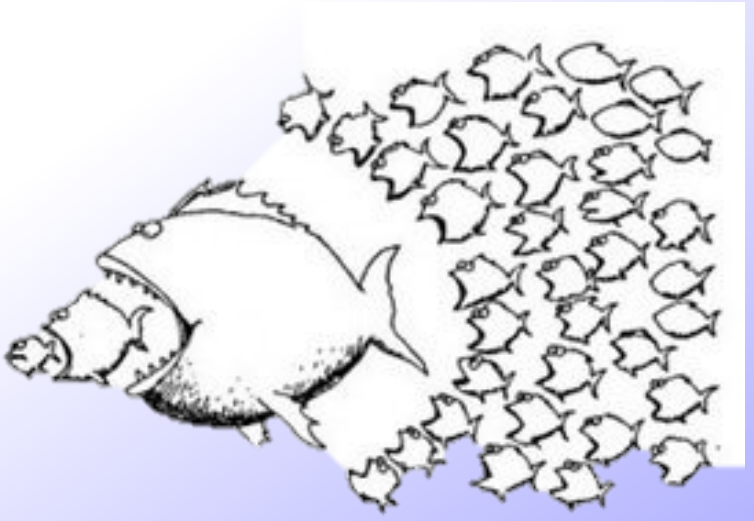
- The internet, and little more

Emergence

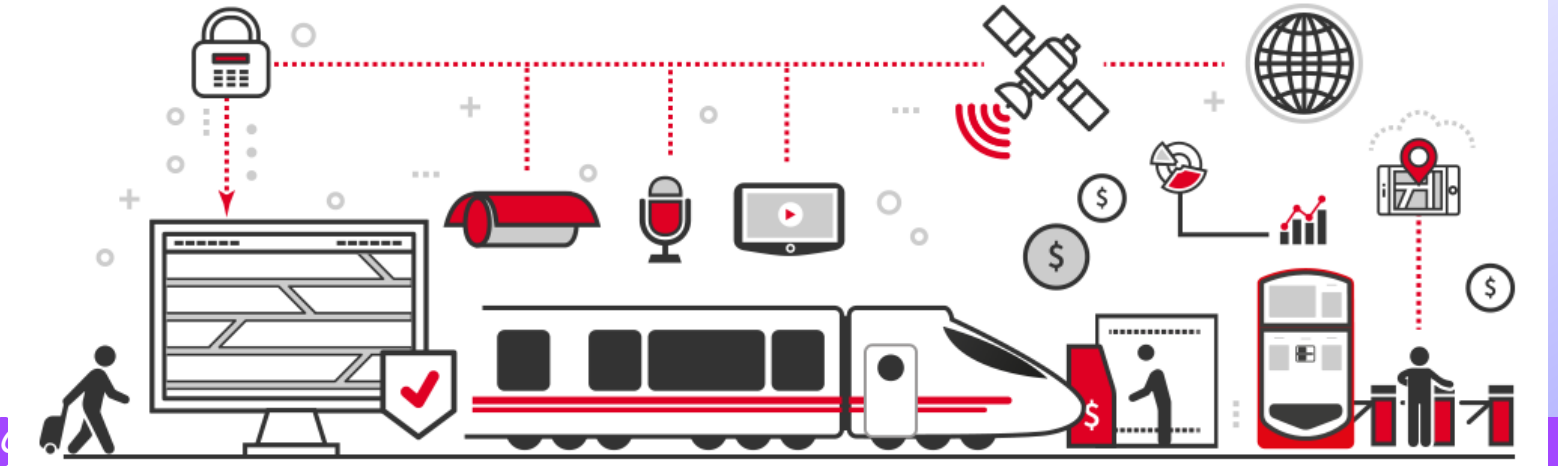
We defined an SoS as an integration of existing (either cyber of physical) subsystems. However, the SoS is not just the sum of its components.

> Emergence: a phenomenon of a whole at the macro-level is emergent if and only if it is of a new kind with respect to the non-relational phenomena of any of its proper parts at the micro level.

- Emergent phenomena can be
  - either beneficial or detrimental, and
  - either expected or unexpected.
- Managing emergence is
  - Essential to avoid undesired, possibly unexpected situations
  - Usually the higher goal of an SoS



Emergence: some examples (for now, we address them in an intuitive way)



Managing Time

Notion of Time

In the (Cyber-Physical) SoS paradigm we start being concerned with change, that depends on the progression of time.

In an SoS a global notion of time is required in order to:

- Enable the interpretation of timestamps in the different CSs.
  - E.g.: ?
- Limit the validity of real-time control data.
  - E.g.: yellow traffic lights.
- Synchronize input and output actions across nodes.
- Provide conflict-free resource allocation.
- Perform prompt error detection.
- Strengthen security protocols.
  - E.g.:?

Time Cycle

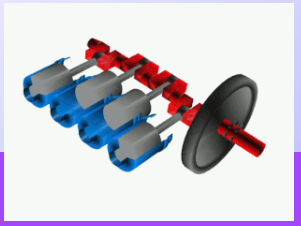
Time: A continuous measurable physical quantity in which events occur in a sequence proceeding from the past to the present to the future.

Timeline: A dense line denoting the independent progression of time from the past to the future.

- Instant: A cut of the timeline (no duration).
- Event: An happening at an instant.

Cycle: A temporal sequence of events that arrives at a final state related to the initial state, from which the temporal sequence of events can be re-started

- An example for a cycle is the rotation of a crankshaft in an automotive engine.
- Although the duration of the cycle changes, the sequence of the significant events during a cycle is always the same.



Periodic Systems

Period: A cycle marked by a constant duration between the related states at the start and the end of the cycle.

Periodic System: A system where the temporal behavior is structured into a sequence of periods.

Periodic Systems are of utmost relevance in control applications

Can you guess why, very intuitively?

How would you implement a sense – elaborate- react system?

(e.g., acquire image, elaborate, enact brakes)

Periodic Systems

Period: A cycle marked by a constant duration between the related states at the start and the end of the cycle.

Periodic System: A system where the temporal behavior is structured into a sequence of periods.

Periodic Systems are of utmost relevance in control applications

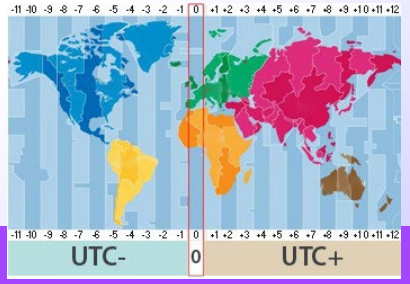
Can you guess why, very intuitively?

Periodicity is not mandatory, but often assumed as it leads to simpler algorithms and more stable and secure systems

- the difference between cycle and period is the constant duration of the period.

Time Standards

- The **physical** second is the same in all UTC, TAI and GPS time standards
- UTC (Universal Time Coordinated) is an astronomical time standard aligned with the rotation of the earth.
  - Since the rotational speed of the earth is not constant, it was decided to base the SI second on atomic processes establishing the International Atomic Time TAI (*Temps Atomique International*).
    - On January 1, 1958 at 00:00:00 TAI and UTC had the same value.
    - TAI is distributed world-wide by the GPS (Global Positioning System) satellites.
  - GPS represents the TAI time in weeks and full seconds within a week.
    - The week count is restarted every 1024 weeks, i.e., after 19.6 years.



Coordinated Clocks

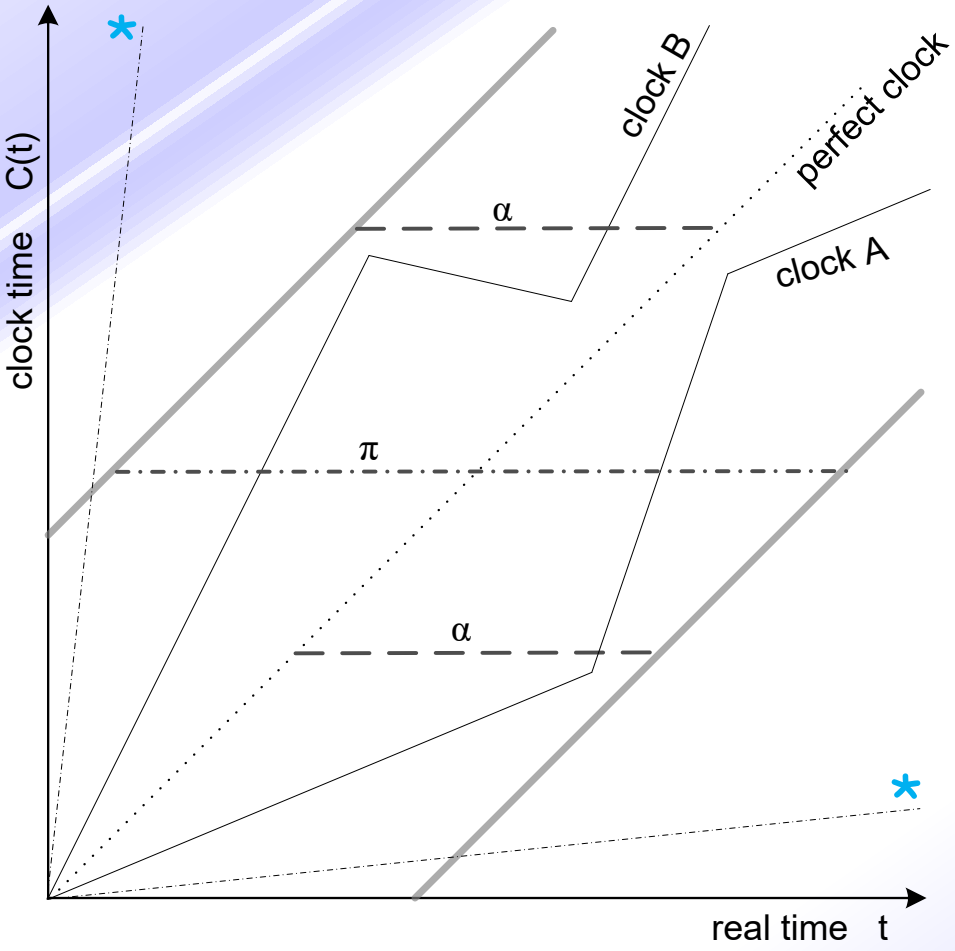
Clock: A (digital) clock is an autonomous system that consists of an oscillator and a register.
Whenever the oscillator completes a period, an event (**tick**) is generated that increments the register.

Reference clock: A *hypothetical clock* of granularity smaller than any duration of interest and whose state is in agreement with TAI.

- the reference clock has small granularity that digitalization errors are neglected,
- the reference clock can observe every event of interest without any delay and
- the state of the reference clock is always in perfect agreement with TAI time.

Coordinated Clock: A clock synchronized within stated limits to a reference clock that is spatially separated

First an example, then definitions (next slides)



A set of two clocks:
precision π,
accuracy α,
drift and
clock synchronization

The outside thick dashed lines represent the bound in the rate of drift*,

*a fundamental assumption for deterministic clock synchronization, since it allows to predict the maximum deviation after a given time interval

Clock Drift

Every good (fault-free) free-running clock has an individual granularity that can be different from the nominal granularity.

Drift: The drift of a physical clock is a quality measure describing the frequency ratio between the physical clock and the reference clock.

- Since the drift of a good clock is a number close to 1, it is conducive to introduce a drift rate by

$$\text{Drift Rate} = |\text{Drift} - 1|$$

- Typical clocks have a drift rate of 10^{-4} to 10^{-8} ppm

Clock synchronization

Precision

The goal is to keep the deviation between two clocks on any two machines within a specified bound, known as the precision π:

$$\forall t,\ \forall p, q : |C_p(t) - C_q(t)| \leq \pi$$

with $C_p(t)$ the computed clock time of machine p at UTC time t.

Accuracy

In the case of accuracy, we aim to keep the clock bound to a value α:

$$\forall t, \forall p: |C_p(t) - t| \leq \alpha$$

Synchronization

- Internal synchronization: keep clocks precise
- External synchronization: keep clocks accurate

Clock drift

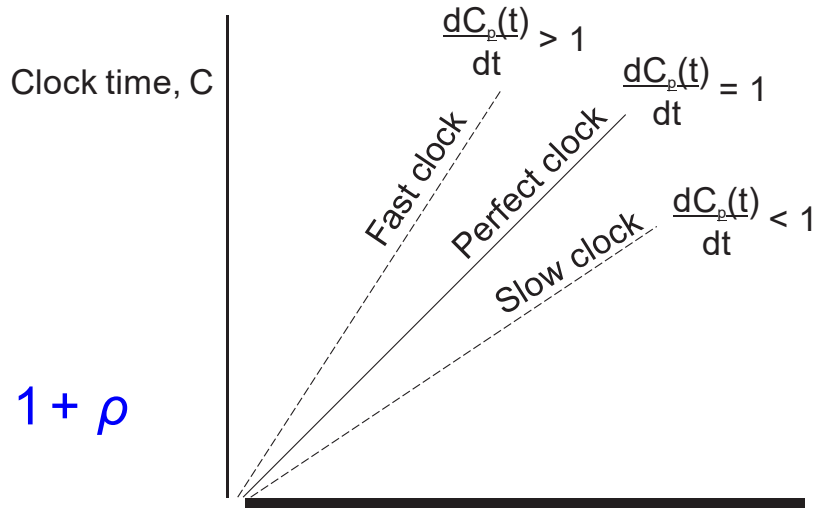
Clock specifications

A clock comes specified with its maximum clock drift rate ρ.
$F(t)$ denotes oscillator frequency of the hardware clock at time t
F is the clock's ideal (constant) frequency $\Rightarrow$ living up to specifications:

$$\forall t: (1-\rho) \leq \frac{F(t)}{F} \leq (1+\rho)$$

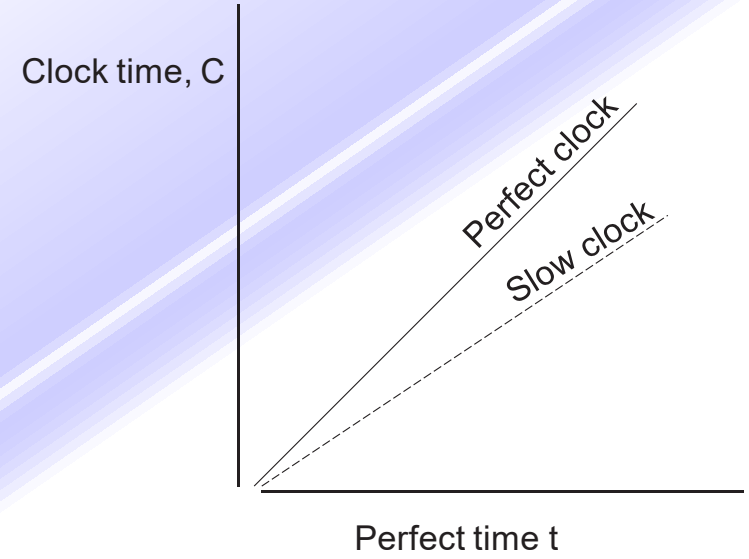
Observation

By using hardware interrupts we couple a software clock to the hardware clock, and thus also its clock drift rate.

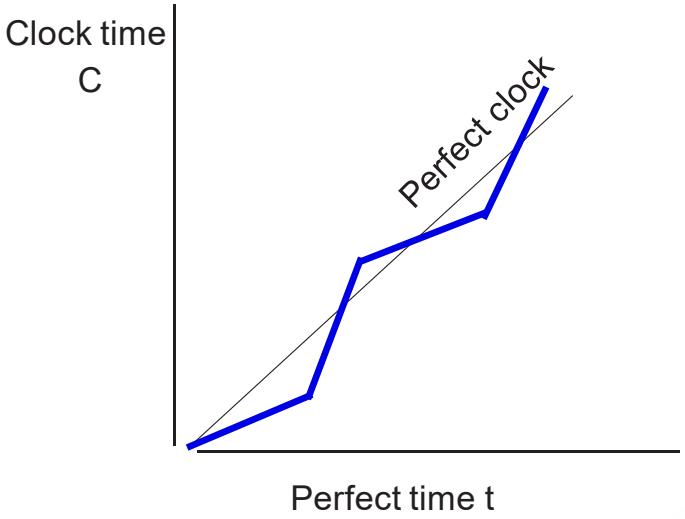


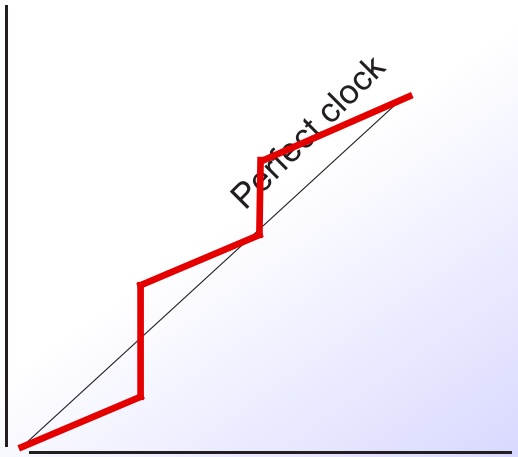
$$\forall t : 1-\rho \leq \frac{dC_p(t)}{dt} \leq 1+\rho$$

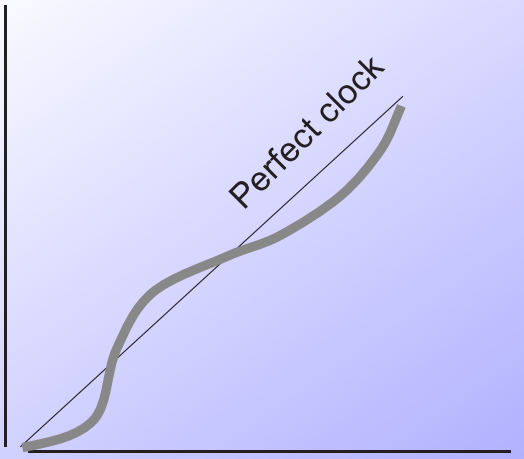
Which disciplining approach should I use? Why?



www.wooclap.com/QDMTNK

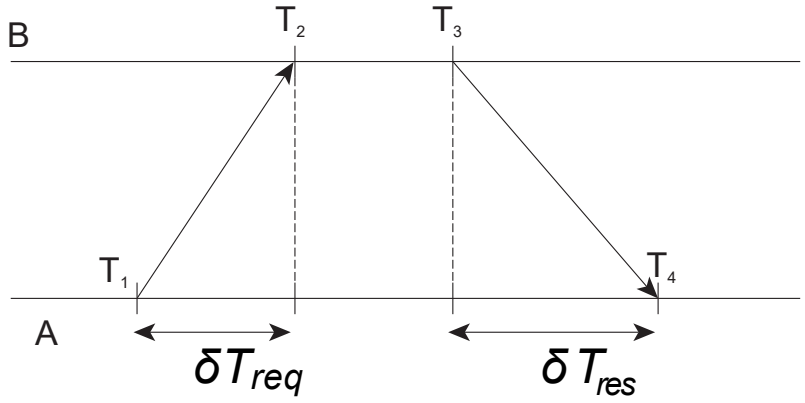






Detecting and adjusting incorrect times

Getting the current time from a time server



Computing the relative offset θ and delay δ

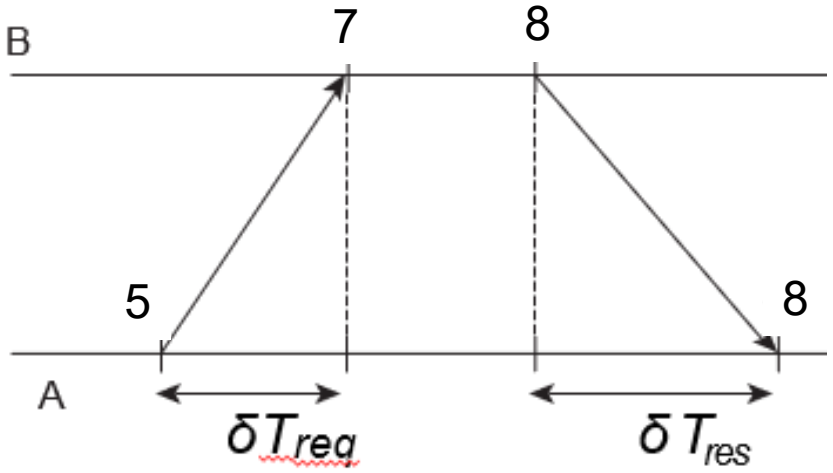
Assumption: $\delta T_{req} = T_2 - T_1 \approx T_4 - T_3 = \delta T_{res}$

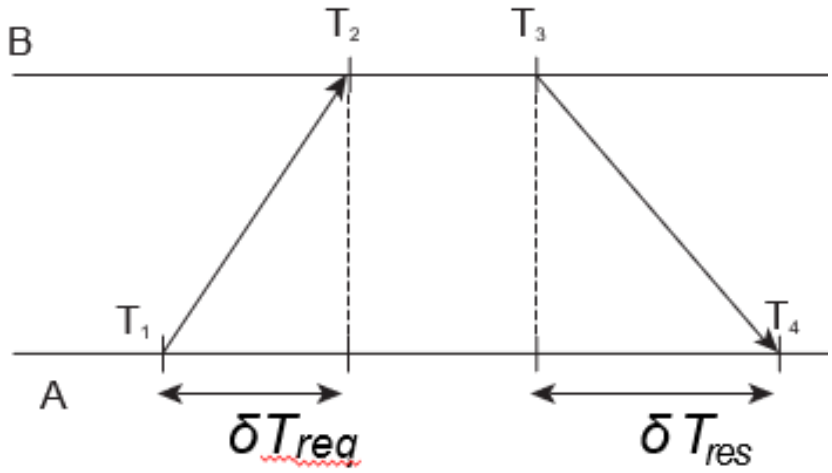
$$\theta = T_3 + ((T_2 - T_1) + (T_4 - T_3)) / 2 - T_4 = ((T_2 - T_1) + (T_3 - T_4)) / 2$$

$$\delta = ((T_4 - T_1) - (T_3 - T_2)) / 2$$

Detecting and adjusting incorrect times

Getting the current time from a time server





Computing the relative offset θ and delay δ

Assumption: $\delta T_{req} = T_2 - T_1 \approx T_4 - T_3 = \delta T_{res}$

$$\theta = T_3 + ((T_2 - T_1) + (T_4 - T_3)) / 2 - T_4 = ((T_2 - T_1) + (T_3 - T_4)) / 2$$

$$\delta = ((T_4 - T_1) - (T_3 - T_2)) / 2$$

Data and State

Data and Information

Systems-of-Systems (SoSs) come about by the **transfer of information** of one Constituent System (CS) to another CS.

- But what is information? How is information related to data?

Data and Information

Systems-of-Systems (SoSs) come about by the **transfer of information** of one Constituent System (CS) to another CS.

- But what is information? Is it any different from data?

Data: A data item is an artefact, a pattern, created for a specified purpose.

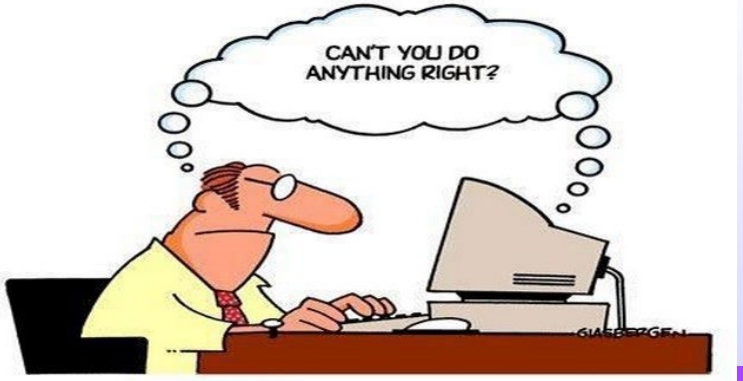
- In cyber space, data is represented by a **bit-pattern**. To expand the meaning of the bit pattern we need to understand how to interpret the given bit pattern.

Information: A proposition about the state of or an action in the world.

Data Receivers

Such data can be intended either for a receiver human or a machine

- Human Receiver
  - the explanation must describe the data using concepts that are familiar to the intended human receiver.
- Machine
  - the computer instructions tell the computer system how the data bit-string is partitioned and how they have to be stored, retrieved, and processed.
    - bit-string: data; how to use: meta-data. Example: XML
  - the explanation of purpose is directed to humans who are involved in the design and operation of the SoS. Therefore, it should be understandable to the user/designer. Example: measurement units.

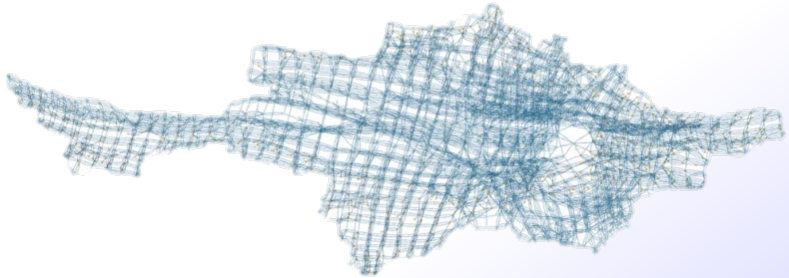


State

- Systems define their behavior depending on interactions with the environment

State: The state of a system at a given instant is the totality of the information from the past that can have an influence on the future behaviour of a system.

- It is a data structure that characterizes the condition of a system at a given time.
- The concept of state is meaningless without a concept of time, since the distinction between past and future is only possible if the system is time-aware.
  - The variables that hold the stored state in a **state-full system** are **state variables**.



State Space: The state space of a system is formed by the totality of all possible values of the state variables within the domain

Actions and Behaviour

Event-Based View

We can observe the dynamics of a system that consists of discrete variables by an event-based view or by a state-based view.

Event-based view:

- we observe the value of relevant state variables at the beginning of the observation
- record all events (i.e., changes of the state variables)
- observe the time of occurrence of the events in a trace.
- the value of all state variables at any past instant is defined by the recorded trace.

➢ However, if the number of events that can happen is not bounded, the amount of data generated by the event-based view **cannot be bounded**.

Example: car purchase system
https://en.wikipedia.org/wiki/Event-driven_architecture

A consumer purchases a car

- the **car's state changes** from "for sale" to "sold".
  (events cause state changes)

A car dealer's system architecture may treat this state change as an **event whose occurrence can be made known to other applications within the architecture**.

What is produced, published, propagated, detected or consumed is a (typically asynchronous) message called the **event notification**

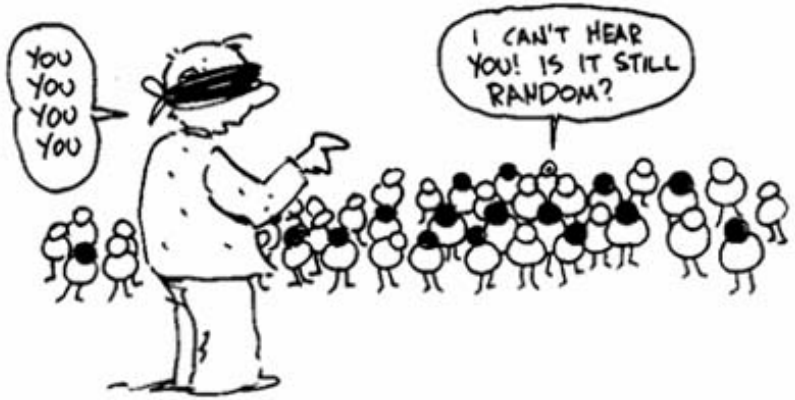
- not the event itself, which is the state change that triggered the message emission. **Events do not travel, they just occur.**

State-Based View

Periodic State-based View (**Sampling**)

- We observe the values of relevant **state variables** at selected observation instants (the **sampling points**) and record these values of the state variables in a **trace**.
- The sampling interval is critical for acquiring a satisfying image of the system.
- The duration between two observation instants puts a **limit on the amount of data** generated by the state-based view.
- Price to pay: events that happen between consequent samples may get lost.

Sampling: The observation of the value of relevant state variables at selected observation instants.



Can you discuss some cases…

Axle counter

www.wooclap.com
QDMTNK

Event-based or state-based?

Consider the observers and their goals:

- A human
- An automatic system

Execution Times (TT and ET)

Execution Time: The time needed to execute a specific action on a system.

- The execution time depends on the performance of the available hardware and is also data dependent.

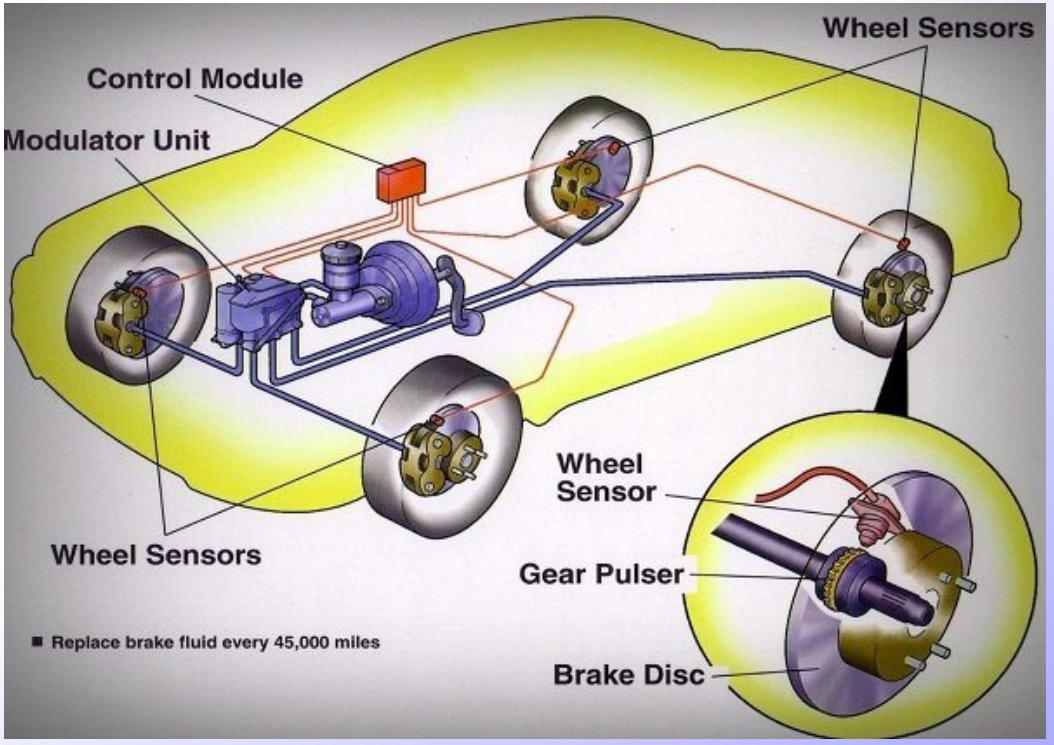
Worst Case Execution Time (WCET): The worst-case data independent execution time required to execute an action on a given computer.

There are two possible sources for a start signal of an action.

- Time-triggered (TT) Action: An action where the start signal is derived from the progression of time.
- Event-triggered (ET) Action: An action where the start signal is derived from an event other than the progression of time.

More on WCET

- BCET - Best-Case Execution Time
- ACET - Average-Case Execution Time
- WCET - Worst Case Execution Time



Event Triggered vs Time triggered

Are the following event-triggered or time-triggered?

- Answering phone calls
- Receiving incoming mails on the mail client
- UNIX signals
- Ethernet with CSMA/CD
  - It uses carrier-sensing to defer transmissions until no other stations are transmitting
- Ethernet with TDMA
  - several users to share the same frequency channel by dividing the signal into different time slots

www.wooclap.com → QDMTNK

it's what we want from a system

Behaviour

The behaviour of a system is of utmost interest to a user.

- Function: A function is a mapping of input data to output data.
- Behaviour: The timed sequence of the effects of input and output actions that can be observed at an interface of a system.

A writing action and a producing output action have an observable effect.

Deterministic Behaviour: A system behaves deterministically if, given an initial state at a defined instant and a set of future timed inputs, the future states, the values and instants of all future outputs are entailed.

A system may exhibit an **intended** or an **erroneous** behaviour.

Service: The intended behaviour of a system.

Communication

Targets of Communication

- A communication system transports a message from a sender to one or more receivers within a given duration and with high dependability.
- By **high dependability** we mean that **by the end of a specified time window**
  - the message should have arrived at the receivers with a high probability
  - the message is not corrupted, either by unintentional or intentional means
  - the security of the message has not been compromised, and that
  - there might be other constraints (e.g., minimal energy consumption)

Communication Protocol: The set of rules that govern a communication action.

Messages

Message: A data structure that is formed for the purpose of the timely exchange of information among computer systems

- Note: a message combines the value domain and the temporal domain

- **Datagram** : A best effort message for the transmission of sporadic messages.
- **PAR-Message** : A PAR-Message (Positive Acknowledgment or Retransmission) is an error controlled transport service for the transmission of sporadic messages from a sender to a single receiver.
- **TT-Message**: A TT-Message (Time-Triggered) is an error controlled transport service for the transmission of periodic messages from a sender to many receivers where the send instant is derived from the progression of the global time.

Match messages type and protocols

- Datagram, PAR-Messages, TT-Messages
- TCP, TDMA, UDP
- What is the correct matching?

www.wooclap.com
QDMTNK

Messages

Message: A data structure that is formed for the purpose of the timely exchange of information among computer systems

- Note: a message combines the value domain and of the temporal domain

- **Datagram** : A best effort message for the transmission of sporadic messages. **(Example protocol: UDP)**
- **PAR-Message** : A PAR-Message (Positive Acknowledgment or Retransmission) is an error controlled transport service for the transmission of sporadic messages from a sender to a single receiver. **(Example protocol: TCP)**
- **TT-Message**: A TT-Message (Time-Triggered) is an error controlled transport service for the transmission of periodic messages from a sender to many receivers where the send instant is derived from the progression of the global time. **(Example protocol: TDMA)**

Comparison of Messages

Characteristic	Datagram	PAR-message	TT-message
Send Instants	Sporadic	Sporadic	Periodic
Data/Control Flow	Uni-directional	Bi-directional	Uni-directional
Flow Control	None	Explicit	Implicit
Transport Duration	A-priori Unknown	Upper-limit Known	Tight-limit Known
Message Jitter	Unknown	Large	Small
Temporal Error Detection	None	At Sender	At Receiver
Example	UDP	TCP	TT-ethernet TDMA

Stigmergy (I)

Constituent systems (CSs) that form the autonomous subsystems of SoSs can exchange information items via two different types of channels:

- the conventional communication channels for the transport of messages and
- the stigmergic channels that transport information via the change and observation of states in the environment.

Stigmergy: it is a mechanism of indirect coordination between agents or actions. The principle is that the trace left in the environment by an action stimulates the performance of a next action, by the same or a different agent.

we have communication!

Stigmergy (II)

The concept of stigmergy has been first introduced in the field of biology to capture the indirect information flow among ants working together.

- Whenever an ant builds or follows a trail, it deposits a greater or lesser amount of pheromone on the trail, depending on whether it has found a prey or not.
- If a prey is found, successful trails end up with a high concentration of pheromone.
- The speed of the ants on a trail is a function of the pheromone concentration.
- Since the trail-pheromone evaporates (we call this process environmental dynamics) unused trails disappear autonomously as time progresses.

Environmental Dynamics: Autonomous environmental processes that cause a change of state variables in the physical environment

List the stigmergic channels

www.wooclap.com/QDMTNK

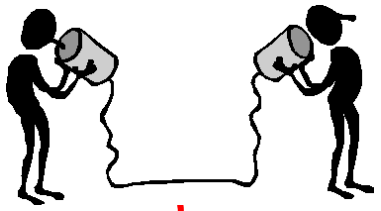
A typical application

Interfaces

Interactions over Channels

Central to the integration of systems are their interfaces

- points of interaction between systems and the environment over time.
- a channel represents this exchange of information at connected interfaces.



Interaction: An interaction is an exchange of information at connected interfaces.

Channel: A logical or physical link that transports information among systems at their connected interfaces.

A channel is implemented by a communication system

- e.g., a computer network, or a physical transmission medium
- affecting the transported information, e.g., by introducing uncertainties
- a channel model describes all channel effects relevant to the transfer of information (BER, bandwidth, ...)

RUIs

RUI: an **interface** of a CS where the services are offered to other CSs.

- the SoS as a whole relies on the services provided by the CSs across the RUIs.

Relied upon Interface (RUI): An interface of a CS where the services of the CS are offered to other CSs.

Relied upon Message Interface (RUMI)

A message interface where the services of a CS are offered to the other CSs of an SoS.

Relied upon Physical Interface (RUPI)

A physical interface where things or energy are exchanged among the CSs of an SoS.

Relied upon Service (RUS)

(Part of) a Constituent System (CS) service that is offered at the Relied Upon Interface (RUI) of a service providing CS under a Service Level Agreement (SLA).

Other Interfaces

Time-Synchronization Interface (TSI): The TSI enables external time-synchronization to establish a global time-base for time-aware CPSoSs.

Utility Interface: An interface of a CS that is used for the configuration, the control, or the observation of the behaviour of the CS.

- The purposes of the utility interfaces are to
  - configure, diagnose and update the system,
  - let the system interact with its physical environment.
- As example, we introduce
  - Diagnosis Interface (D-Interface): An interface that exposes the internals of a Constituent System (CS) for the purpose of diagnosis. Example: car ECU's fault codes.
  - Monitoring CS: A CS of an SoS that monitors the information exchanges across the RUMIs of an SoS or the operation of selected CSs across the D-Interface. Examples: from IDS to telemetry.

Evolution and Dynamicity

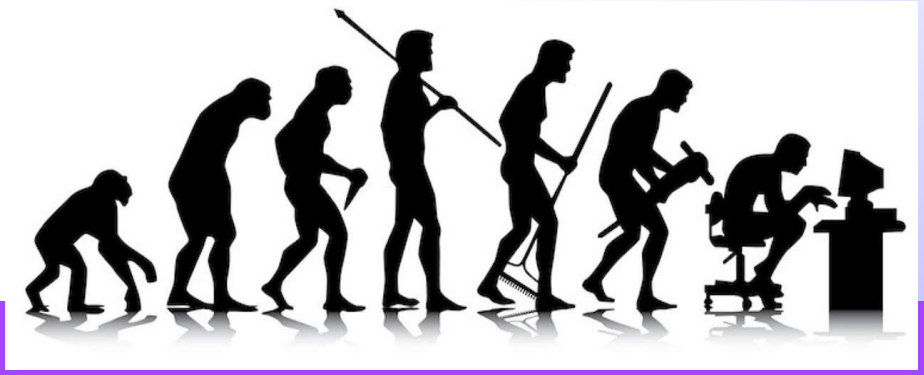
Dynamicity and Evolution

Large scale Systems-of-Systems are designed for a long period of usage

- Over time, the demands and the constraints put on the system will usually change, as will the environment in which the system is to operate.
- Short-Term changes are referred as Dynamicity
- Long-Term (planned) changes are referred as Evolution

Dynamicity: The capability of a system to react promptly to changes in the environment

Evolution: Process of gradual and progressive change or development, resulting from changes in its environment (primary) or in itself (secondary).



Example of Dynamicity: Smart Grids

- Prosumer/consumers
- Energy on demands
- Load balancing

SoS Evolution

Although the term evolution in other contexts does not have a positive or negative direction, in SoSs evolution refers to maintaining and optimizing the system.

- More in detail, evolution is needed to cope with changes:
  - **Managed evolution** refers to the evolution guidance
  - The goal can be anything like performance , efficiency , etc.

- **Managed SoS evolution**: Process of modifying the SoS to keep it relevant in face of an ever-changing environment.
- **Unmanaged SoS evolution**: Ongoing modification of the SoS that occurs as a result of ongoing changes in (some of) its CSs.

Cases of managed and unmanaged evolution

Reconfigurability

The capability of a system to adapt its internal structure in order to mitigate internal failures or to improve the service quality.

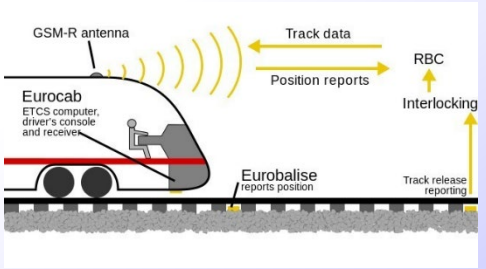
- Linked to dynamicity
- Example: Graceful degradation in telecom network

SoS Authority

Sometimes, governance-related facts may have impact on SoS evolution. Governance is generally related to:

Authority: The relationship in which one party has the right to demand changes in the behaviour or configuration of another party, which has to conform to them.

(Collaborative) SoS Authority: An organizational entity that has societal, legal, and/or business responsibilities to keep a collaborative SoS relevant to its stakeholders. To this end it has authority over RUI specifications and how changes to them are rolled out.



System Design and Tools

SoS Architecture

Sos can have a very complex architecture

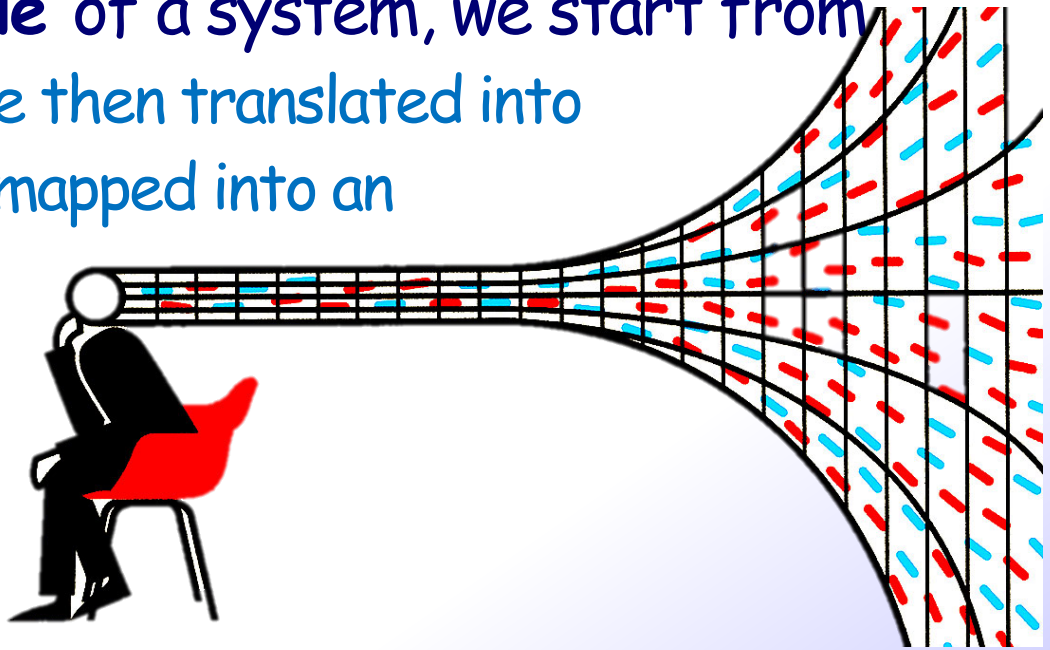
- **Problem**: some SoS requirements may not be fulfilled.
- **Solution**: The architecture of a system can have some variants or even can vary during its operation.
  - **Evolvable architecture**: it is adaptable and then is able to incorporate known and unknown changes in the environment or in itself.
  - **Flexible architecture**: it can be adapted to a variety of future possible developments.
  - **Robust architecture**: it performs well under a variety of possible future developments.

- The architecture then involves several components which interact with each other through interfaces.

Design Process

During the **development lifecycle** of a system, we start from

- **conceptual thoughts** which are then translated into
- **requirements**, which are then mapped into an
- **architecture**.



Design: The process of defining an architecture, components, modules and interfaces of a system to satisfy specified requirement.

Modularity: Engineering technique that builds larger systems by integrating modules.

Design Evolving Systems

Design for evolution: Exploration of forward compatible system architectures, i.e. designing applications that can evolve with an ever-changing environment.

- Design for evolution aims to achieve robust and/or flexible architectures.
- Principles of evolvability include modularity, updateability and extensibility.

In the context of SoS, design for evolution means that **expected** changes should be accommodated without any global impact on the architecture.

- 'Expected' refers to the fact that **changes will happen**, it does not mean that these changes themselves are foreseeable.

Design for testability: The architectural and design decisions in order to enable easy and effective testing of the system.

Dependability and Security

Introduction: Concepts and Definitions

Basic **Dependability** Concepts

Definitions:

- **Faults, Errors and Failures**
- Dependability Attributes

Means to attain dependability

- Fault Prevention
- Fault Tolerance
- Fault Removal
- Fault Forecasting

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

AMA-CPS

Dependability and security: added value or a commodity?

Our society depends more and more on the proper behavior of **cyber-physical computing systems**.

They are used for critical services, very often overcoming human intervention, either because of an explicit design choice or because humans have unacceptable reaction time.

Some examples - 1

CNN tech BUSINESS CULTURE GADGETS FUTURE STARTUPS

Why do computer crashes keep bringing major airlines to their knees, leaving hundreds of thousands of passengers stranded at airports? Human error. Mistakes. Good old fashioned screw ups.

That's the explanation offered by airline computer experts Monday, after Delta Air Lines scrambled to deal with a huge computer snafu. The world's second largest airline was forced to delay of all its flights on the ground for at least six hours worldwide.

Delta (DAL) blamed the problem on a power outage. But that alone should not have brought the system down -- there are backups that should have kept Delta's system up and running.

"You're basically saying, 'We had power failure in a location but unfortunately we were unable to continue operations from a secondary data center despite the fact that we spent hundreds of millions of dollars on it.," said Gil Hecht, founder and chief executive of Continuity Software, an expert in computer disaster recovery.

That is essentially an admission of human error, added Hecht.

Nest Thermostat Glitch Leaves Users in the Cold

Disruptions
By NICK BILTON JAN. 13, 2016

The Nest Learning Thermostat is dead to me, literally. Last week, my once-beloved "smart" thermostat suffered from a mysterious software bug that drained its battery and sent our home into a chill in the middle of the night.

Although I had set the thermostat to 70 degrees overnight, my wife and I were woken by a crying baby at 4 a.m. The thermometer in his room read 64

HSBC to process 275,000 delayed payments overnight

System failure leaves many workers without their salaries ahead of bank holiday weekend, but bank promises to refund any costs incurred

Chaos as global IT failure takes out all British Airways flights out of London

Posted May 27, 2017 by Mike Butcher (@mikebutcher)

Some examples - 2

US prisoners released early by software bug

23 December 2015 | Technology

Share

Software Glitch Causes F-35 to Incorrectly Detect Targets in Formation

Toyota to Pay $1.2B for Hiding Deadly 'Unintended Acceleration'

ABC News first reported concerns in 2009; FBI: Toyota "put sales over safety."

plus, security and privacy…

The Heartbleed Bug

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.

RISK ASSESSMENT —

How Hacking Team got hacked

A black hat claims responsibility for the hack. Here's how he says he did it.

J.M. PORUP (UK) - 4/19/2016, 3:36 PM

Infamous examples – Ariane 5 (1996)

Ariane 5 lost due to ***correct*** component from Ariane 4 (approx. $350m damage)

- The internal computers were reused from the Ariane 4
- The greater speed caused the computers to experience an overflow while converting a 64-bit floating point number to a 16-bit integer.
  - Ariane5 exceeded the limits of Ariane4 in just 37 seconds, causing overflow, confusing the rocket and initiating a sudden turn downward that resulted in a catastrophic breakup and aerial explosion.

Infamous examples – Mars Climate Orbiter (1998)

A small space probe launched to enter Martian orbit. Unknowingly, the spacecraft was put on a trajectory that would lead to the failure of the mission.

- The orbiter was being navigated by various teams of people → some used metric units, others imperial units.
- Conversion error—and the misconfiguration of the computer systems— led to a course correction sent the *Mars Climate Orbiter* far too close to the planet, and it was likely violently burned up and destroyed in the atmosphere.

Infamous examples – Knight Capital (2012)

American-based financial services firm buying and sharing stocks of huge value in large quantities on the global stock market.

- One morning, the automated computer systems based at Knight Capital began rapidly buying and selling millions and millions of shares distributed among hundreds of stocks for a total of 45 minutes, before the systems were isolated and stopped.
  - New trading software had been installed improperly on one of the computers by a technician.
- Knight Capital was forced to sell these shares back at low prices, with total net loss of over $440 million.

Knight

Infamous examples – Patriot missile defense system (1991)

Surface-to-air missile system capable of tracking and intercept aircraft and ballistic missiles

A system in Saudi Arabia had been operational for 100 hours, causing its internal clock to **drift by 0.34 seconds.**

- The Israelis had advised the US to periodically reboot the system's computers. This was **not** done.
- A "missile launched by Iraq hit the US Army barracks killing 28 soldiers.

The defence system had activated and detected the missile, predicting where to look for it - **but due to the drift, it looked in the wrong place**

Even more on computer failures

- Radiation therapy machine overdoses patients (Therac-25)
- Buggy recovery software interrupts long-distance telephone service for hours in many states of U.S.A.
- Bank software sends avalanche of spurious order corrections, freezes part of NYSE affecting trading in 975 stocks
- One wrong key stroke dooms Russian satellite

... continuing:

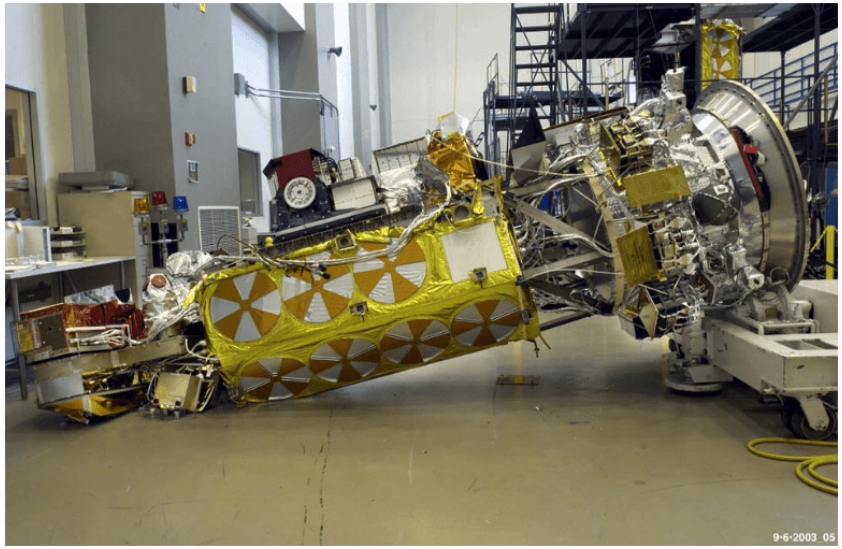
- Several railway Accidents and Killings
- Toyota sudden unintended acceleration
- (2018) Autonomous Car Killing pedestrian

Honorable mention- NOAA-19 Satellite

On September 6, 2003, this satellite was badly damaged while being worked on at the Lockheed Martin Space Systems factory.

- Bug? Glitch? No, it just fell to the floor as a team was turning it to a horizontal position because a lack of procedural discipline throughout the facility.

Repairs cost $135 million.



Dependability

Computing systems are characterized by five fundamental properties: functionality, usability, performance, cost, and dependability.

"Dependability of a computing system is the ability to deliver service that can justifiably be trusted"

(*ability to avoid service failures that are more frequent and more severe than is acceptable*)

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

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

Definitions

Correct service is delivered when the service implements the system function.

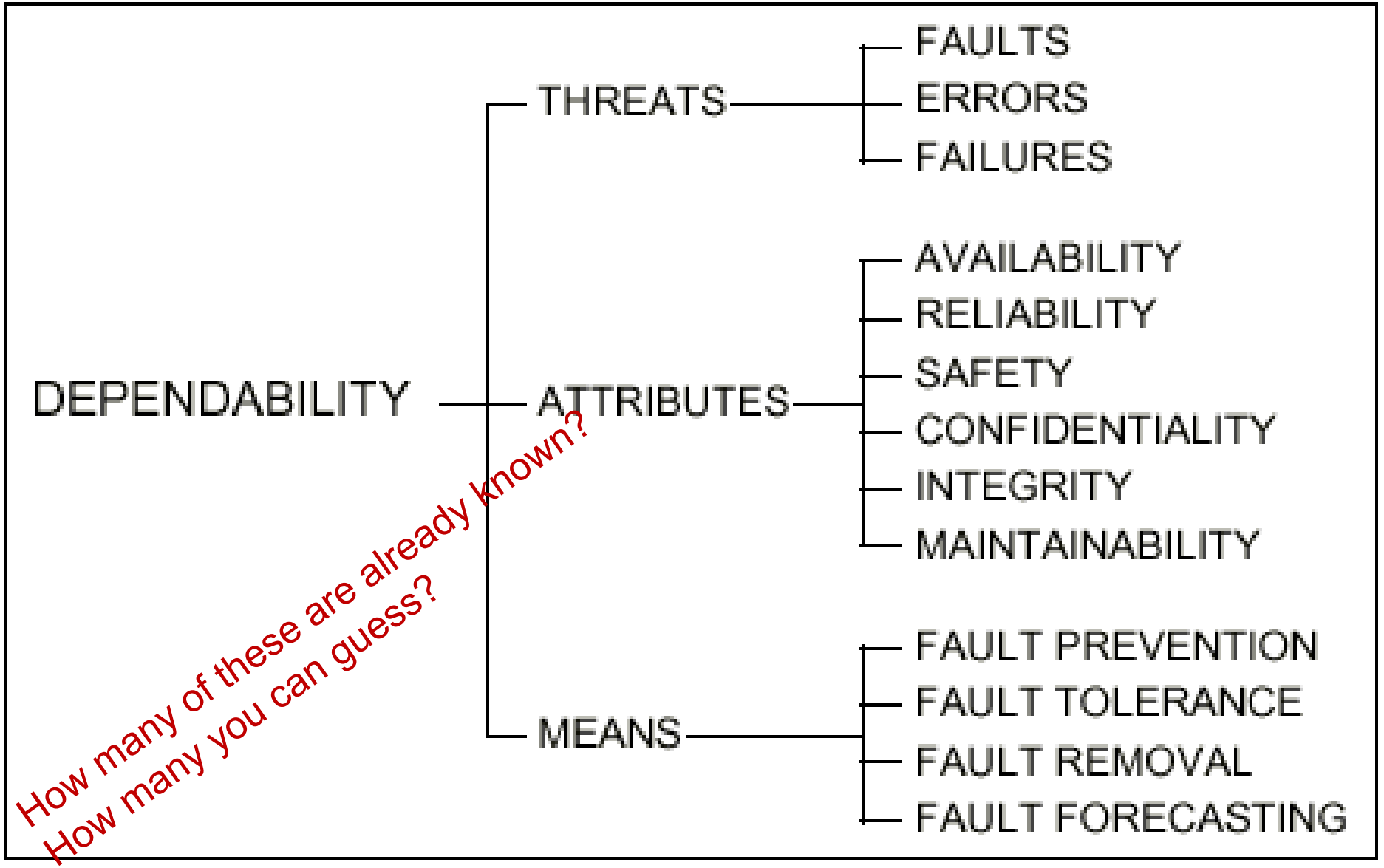
A system failure is an event that occurs when the delivered service **deviates from correct service**.

A failure is thus a transition from correct service to incorrect service, i.e., to not implementing the system function.

The delivery of incorrect service is a **system outage**.

A transition from incorrect service to correct service is service restoration.

An overview
(before we define all of them)



The Threats: Faults, Errors and Failures

A system may fail either because

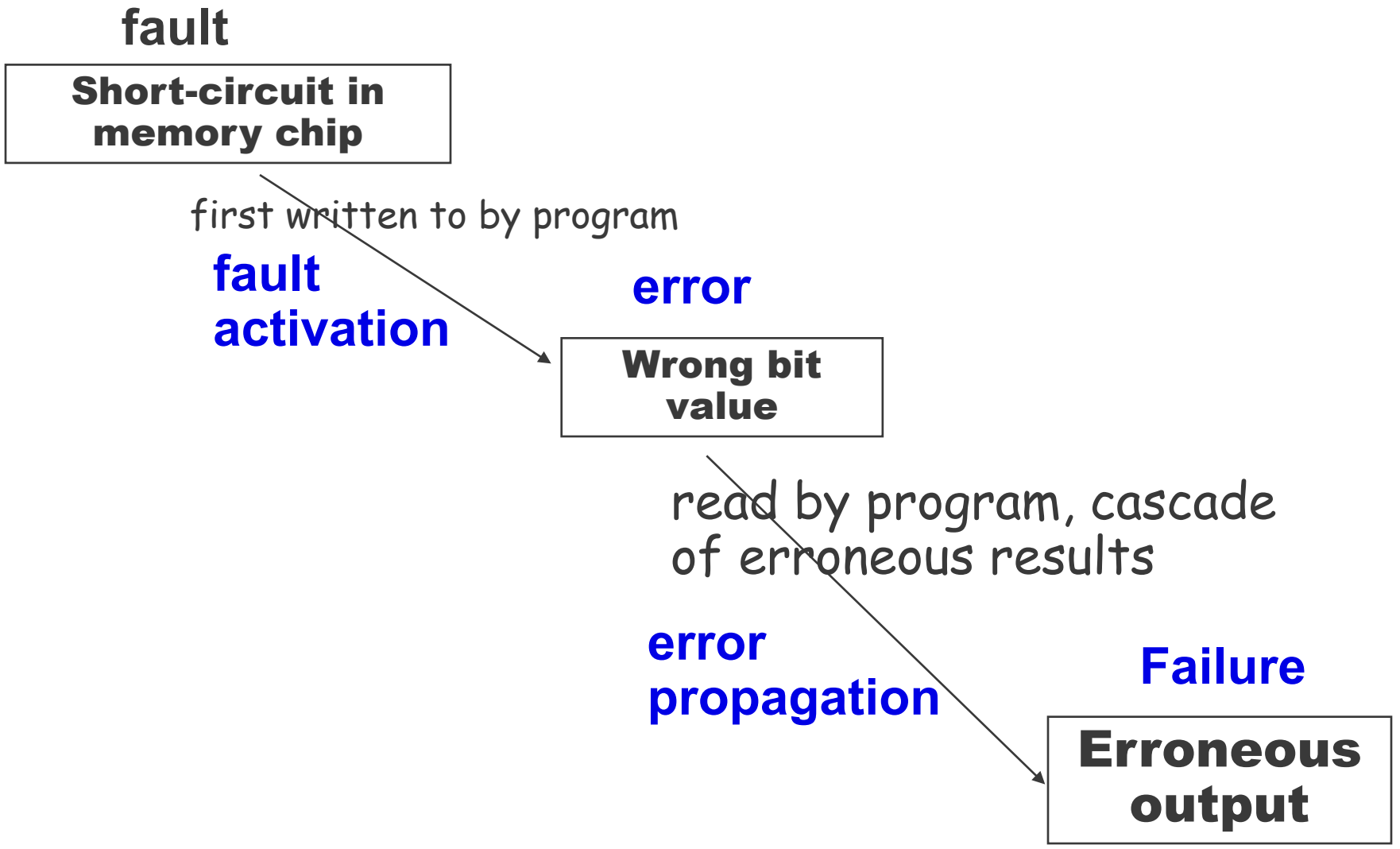
- it does not comply with the specification, or
- the specification did not adequately describe its function.

Error: that part of the system state that may cause a subsequent failure.

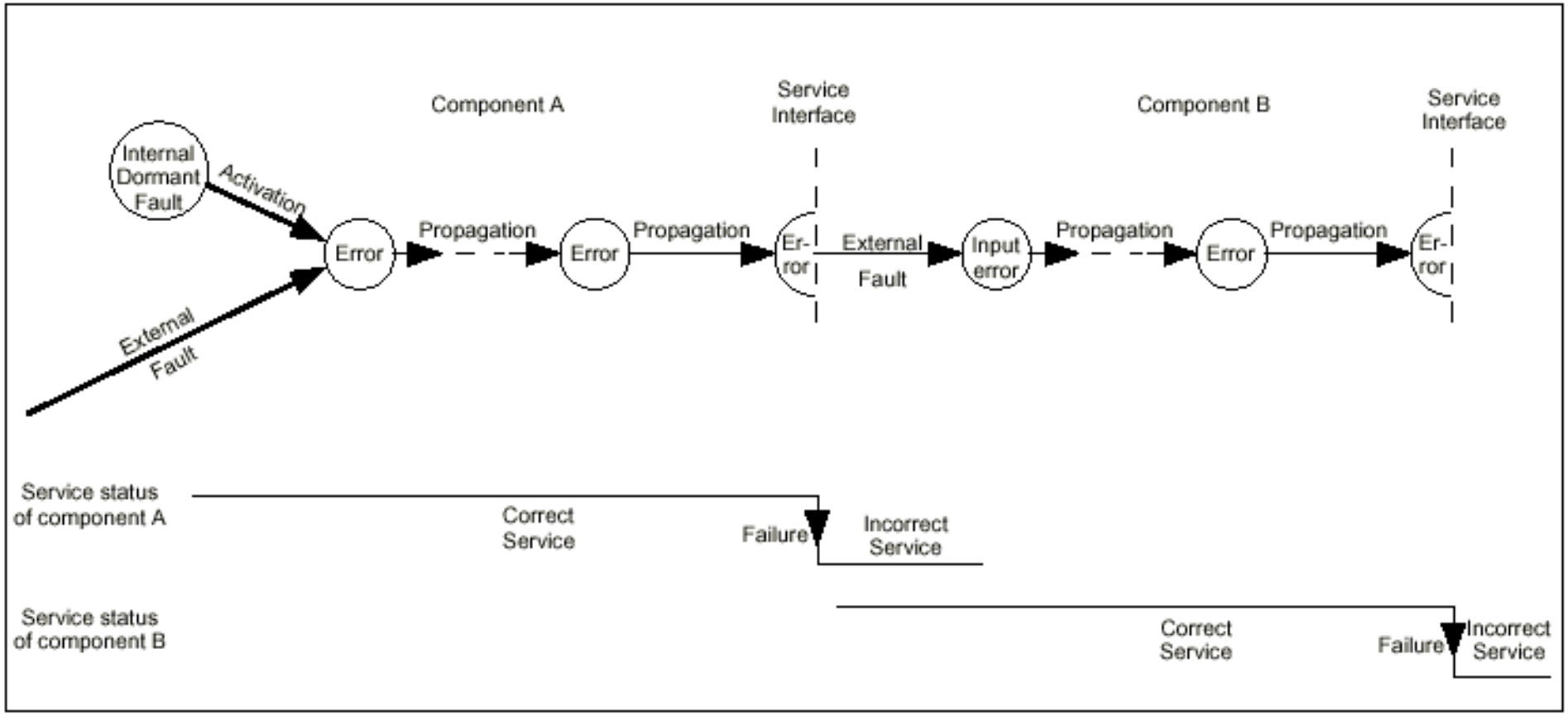
A Failure occurs when an error reaches the service interface and alters the service.

Fault: the adjudged or hypothesized cause of an error. A fault is *active* when it produces an error; otherwise it is *dormant*.

Example of faults-errors-failures



Fault error failure chain - 3



The Attributes of Dependability

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

Availability: readiness for correct service

365 days of service

with 365 minutes of downtime

1 minute of downtime per day

A(365)= 364.7/365

Reliability: continuity of correct service

R(365) → every day it fails

Safety: absence of catastrophic consequences

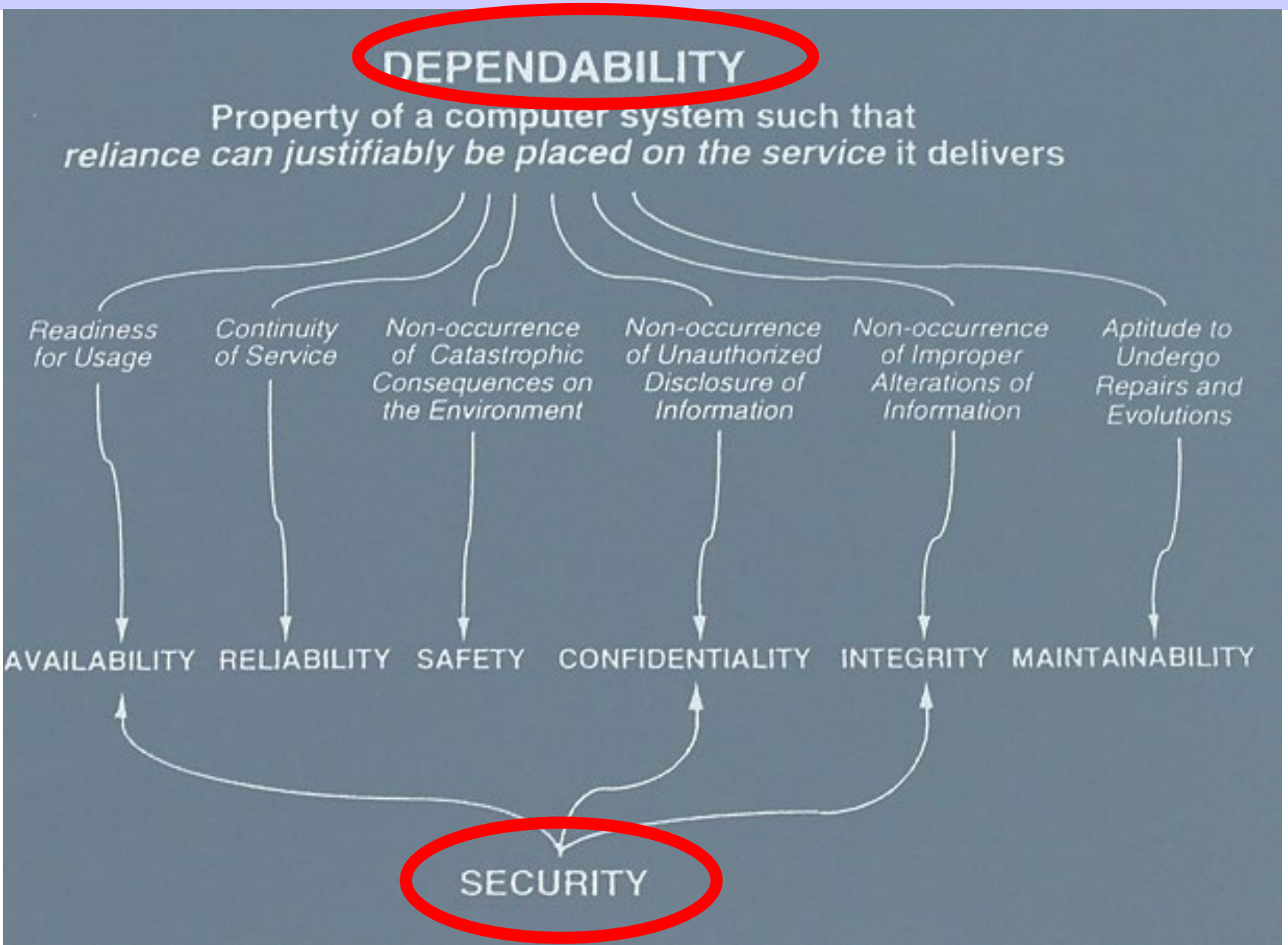
Confidentiality: absence of unauthorized disclosure of information

Integrity: absence of improper system state alterations

Maintainability: ability to undergo repairs and modifications

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

THE ATTRIBUTES OF DEPENDABILITY

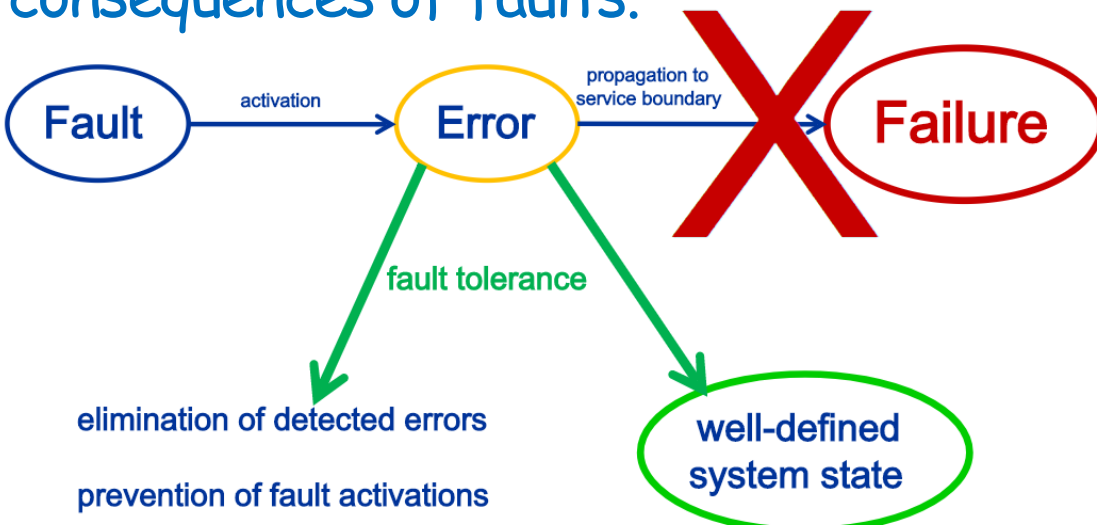


Example

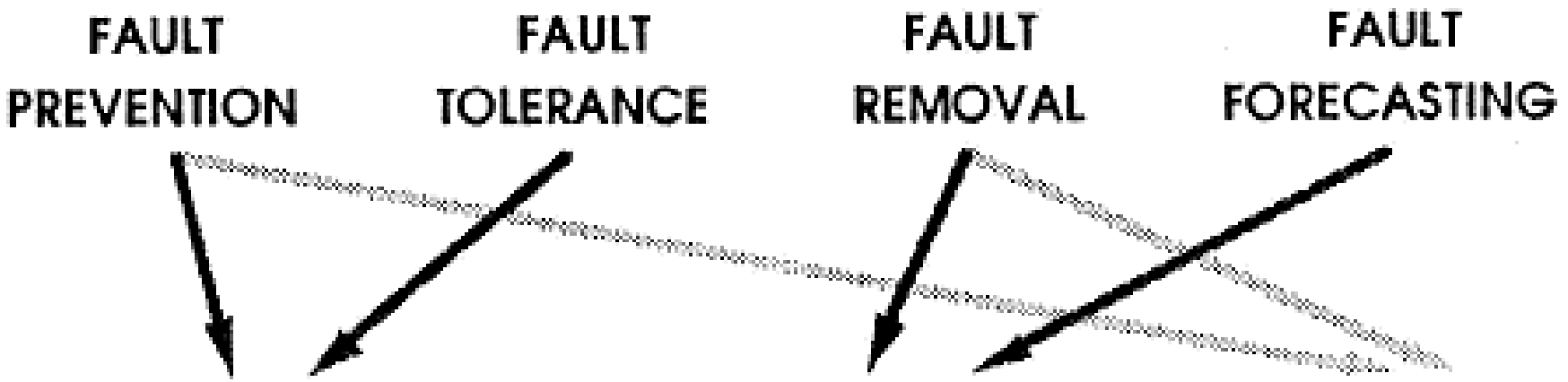
Dependability: Means

The means to attain dependability (and security) are grouped into four major dependability categories :

- **Fault prevention**: The means to prevent the occurrence or introduction of faults.
- **Fault tolerance**: The means to avoid service failures in the presence of faults.
- **Fault removal**: The means to reduce the number and severity of faults.
- **Fault forecasting**: The means to estimate the present number, the future incidence, and the likely consequences of faults.



The Means to Attain Dependability



how to *provide* system with ability to deliver service complying with specification

how to *reach confidence* in system ability to deliver service complying with specification

how to aim at a fault free system

DEPENDABILITY PROCUREMENT

DEPENDABILITY VALIDATION

FAULT AVOIDANCE

Verification and Validation - PMBOK

Validation. The **assurance** that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with **external customers**. Contrast with *verification*.

Verification. The **evaluation** of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an **internal process**. Contrast with *validation*.

Verification and Validation – System and software engineering

Validation

ensure that the right product is built—that is, the product fulfills its specific intended purpos

→ Are we building the right system?

Verification

ensure that the product is built correctly, in the sense that the output products of an activity meet the specifications imposed on them in previous activities

→ Are we building the system right?

V&V constituted by:

- Risk Analysis (HAZOP, FMEA/FMECA), Static Analysis (code inspection, formal methods), Formal Proves, Modelling (FT, RBD, Markov, …) , Simulation, Dynamic Analysis and Testing., …

Security (I)

Security: The composition of confidentiality, integrity, and availability;
Security requires in effect the concurrent existence of availability for authorized actions only, confidentiality, and integrity (with "improper" meaning "unauthorized")

Security allows reducing the risk related to threats e.g., attacks, which may be conducted by external entities who exploit existing vulnerabilities.

Security (II)

Threat: Any circumstance or event with the potential to adversely impact organizational operations / assets / individuals, via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.

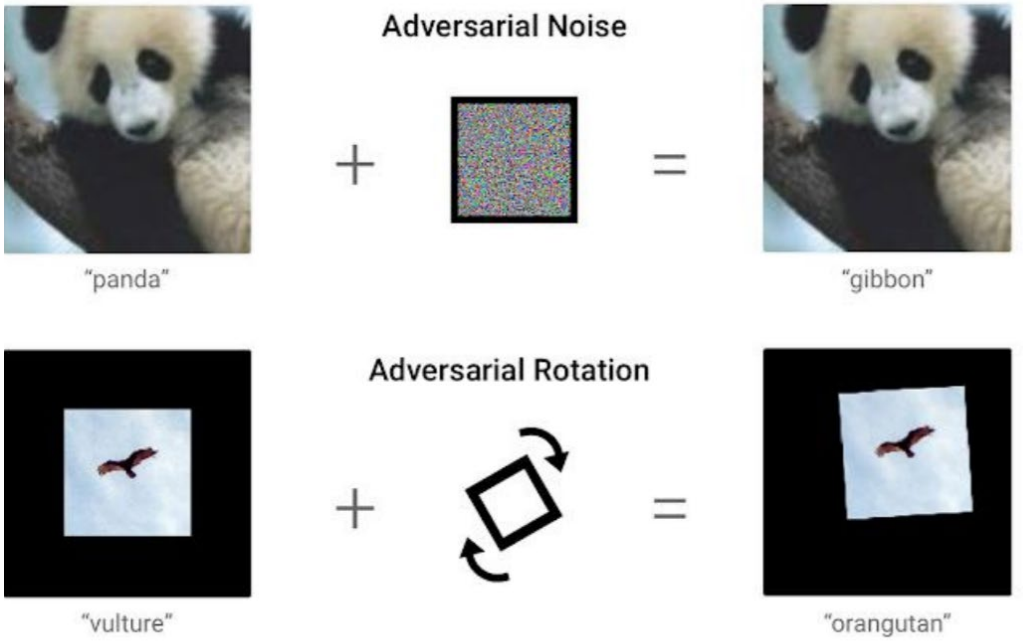
Vulnerability: Weakness in a system, in system security procedures, internal controls, or implementations that could be exploited by a threat.

Risk: A measure of the extent to which an organization is threatened by a potential circumstance or event, and typically a function of

- the adverse impacts that would arise if the circumstance or event occurs;
- and the likelihood of occurrence

Stigmergic channel: old and new attack surfaces?

Smart meters hacking



Brown et al. (2018). Unrestricted Adversarial Examples. *arXiv*

termostate hacking

smart tv

OWASP IoT Attack Surface Areas (2018)

1. Weak Guessable, or Hardcoded Passwords
2. Insecure Network Services
3. Insecure Ecosystem Interfaces
4. Lack of Secure Update Mechanism
5. Use of Insecure or Outdated Components
6. Insufficient Privacy Protection
7. Insecure Data Transfer and Storage
8. Lack of Device Management
9. Insecure Default Settings
10. Lack of Physical Hardening

Discussion – open question

For each example, discuss the main dependability & security requirements