Introduction to Real-Time Industrial Systems

Real-Time Industrial Systems

Marcello Cinque

Introduction to real-time industrial systems

- Roadmap:
  - Basic notions of real-time systems
  - Applications
  - Critical systems and safety
  - Course roadmap

- References:
  - G. Buttazzo – Giorgio Buttazzo: “Hard real-time computing systems: Predictable Scheduling Algorithms and Applications”, Third Edition, Springer, 2011.
    - Chapter 1
  - Avizienis, Laprie, Randwell, Landwehr “Basic Concepts and Taxonomy of Dependable and Secure Computing”. IEEE TDSC 2004

Real-time systems

- Real time systems are computing systems in which correct functioning does not depend only on the validity of results but also on the ***time*** results are produced

- Systems with a double correctness concept
  - Logic (“ it does the right thing”)
  - Temporal (“it does it on time”)

What does *real time* mean?

- The characteristic that distinguish real-time computing from others is ***time***
  - The word *time* indicates that the validity of the results does not depend only on their correctness but also on the time they are obtained
  - The word *real* indicates that the response of the system to external events must happen during the evolvement of those events, thus the internal time of the system has to be measured with a temporal reference that has to be coincident with the one of the environment where the system operates.

Embedded systems

- An **embedded system** is a computer system that has a dedicated function within a larger mechanical or electrical system.
- It is *embedded* as part of a complete device often including electrical or electronic hardware and mechanical parts.
- Because an embedded system typically controls physical operations of the machine that it is embedded within, it often has real-time constraints
- In opposite, **general purpose systems** do not have a dedicated function (e.g., a personal computer)

The role of real-time systems in industry

- Most of the computer systems used in industry have to possess real-time capabilities
  - Supervision control on a manufacturing plant
  - Steer-by-wire or ADAS in cars
  - Cruise control on aircraft
  - Traction control on trains
  - …
- Computer systems tightly integrated (embedded) with the physical system they have to interact with




Deadline

- The deadline is the maximum time in which a real-time task has to terminate its execution
- In a real-time system, a result produced after the deadline (*deadline miss*) can be dangerous
- Depending on the consequence of a deadline miss, real-time systems are classified as *hard* or *soft*

Hard and soft real-time

- **hard real-time task**: a deadline miss might cause catastrophic consequences on the system
  - Sensor acquisition, detection of critical conditions, planning of actions in systems that are strictly tied with the environment, …
- **soft real-time task**: a deadline miss might cause a performance degradation, without jeopardizing the correct functioning of the system
  - Command line interpreter; visualizaion of messages on the monitor, GUI activities, data backup, …

Applications

Multimedia
Streaming
Gaming
...

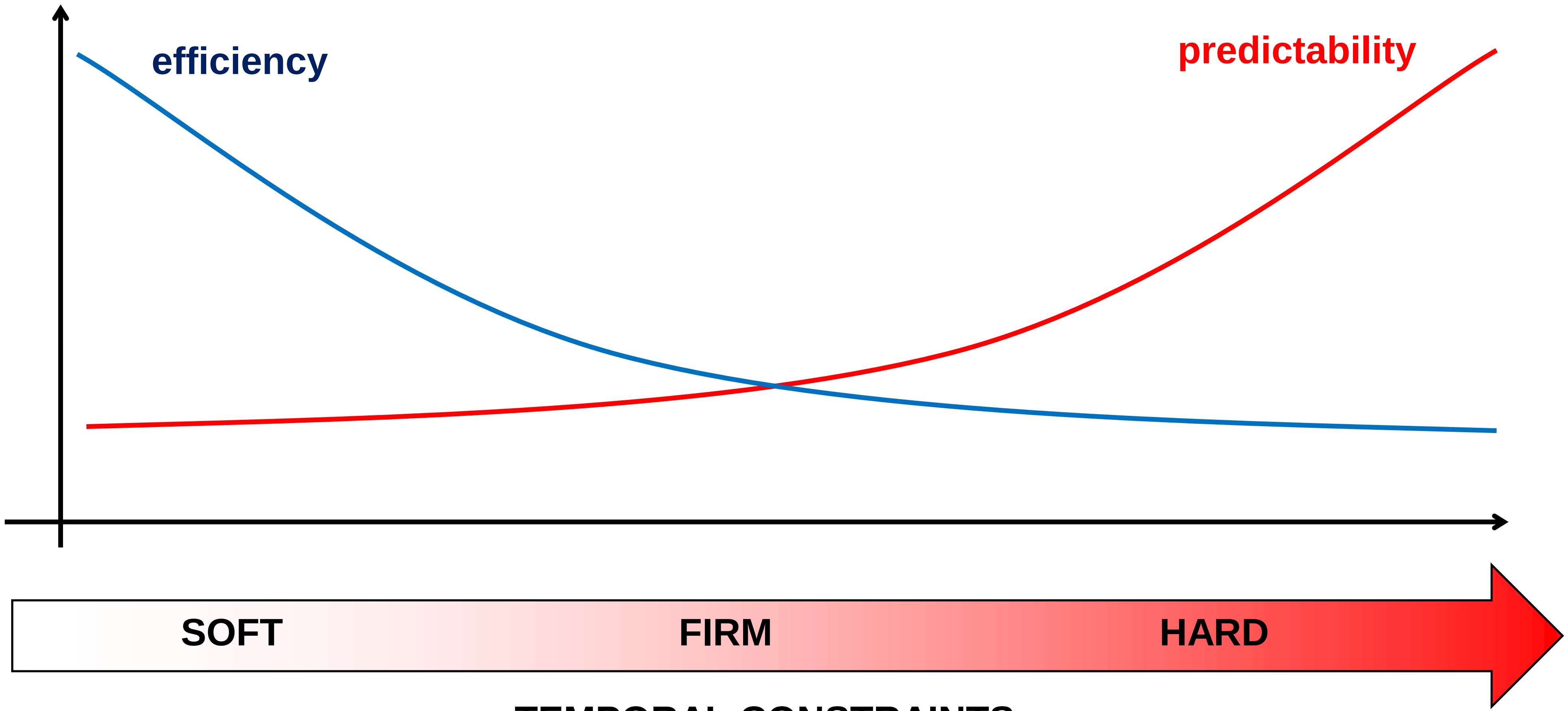
SOFT FIRM HARD

TEMPORAL CONSTRAINTS

Low latency **Safety**

Real-time and speed

- Often, real-time concepts are erroneously confused with the reaction speed of the systems
- Actually, although computer systems' performance continuously improves, this is not sufficient to satisfy temporal constraints
- While the general objective of a quick computation is to minimize the <u>average</u> response time of a task set, **the goal of real-time computation is to satisfy the single, <u>individual</u> temporal needs <u>of each</u> task**



TEMPORAL CONSTRAINTS

Desirable properties of real-time systems

- **Timeliness**: results must be correct both in the logic domain and in the timing domain

- **Predictability**: the system must allow to determine in advance if all its tasks can be terminated within their respective deadlines

- **Overload tolerance:** the system must not collapse in case of excessive load

Desirable properties of real-time systems

- **Monitoring:** the system has to be monitorable, in other terms execution states of tasks must be observable in order to spot deadline misses

- **Flexibility**: the system has to modular and easily changeable, in order to be adapted to application needs

  - And, for hard real-time industrial systems…

- **Safety!**

Critical Systems

Different *critical scenarios*:

- Safety Critical Systems:
  - failure results in loss of life, or damage to the environment
- Mission-critical Systems
  - failure results in failure of some goal-directed activity (aircraft control systems);
- Business-critical systems
  - Failure results in high economic losses (financial systems)

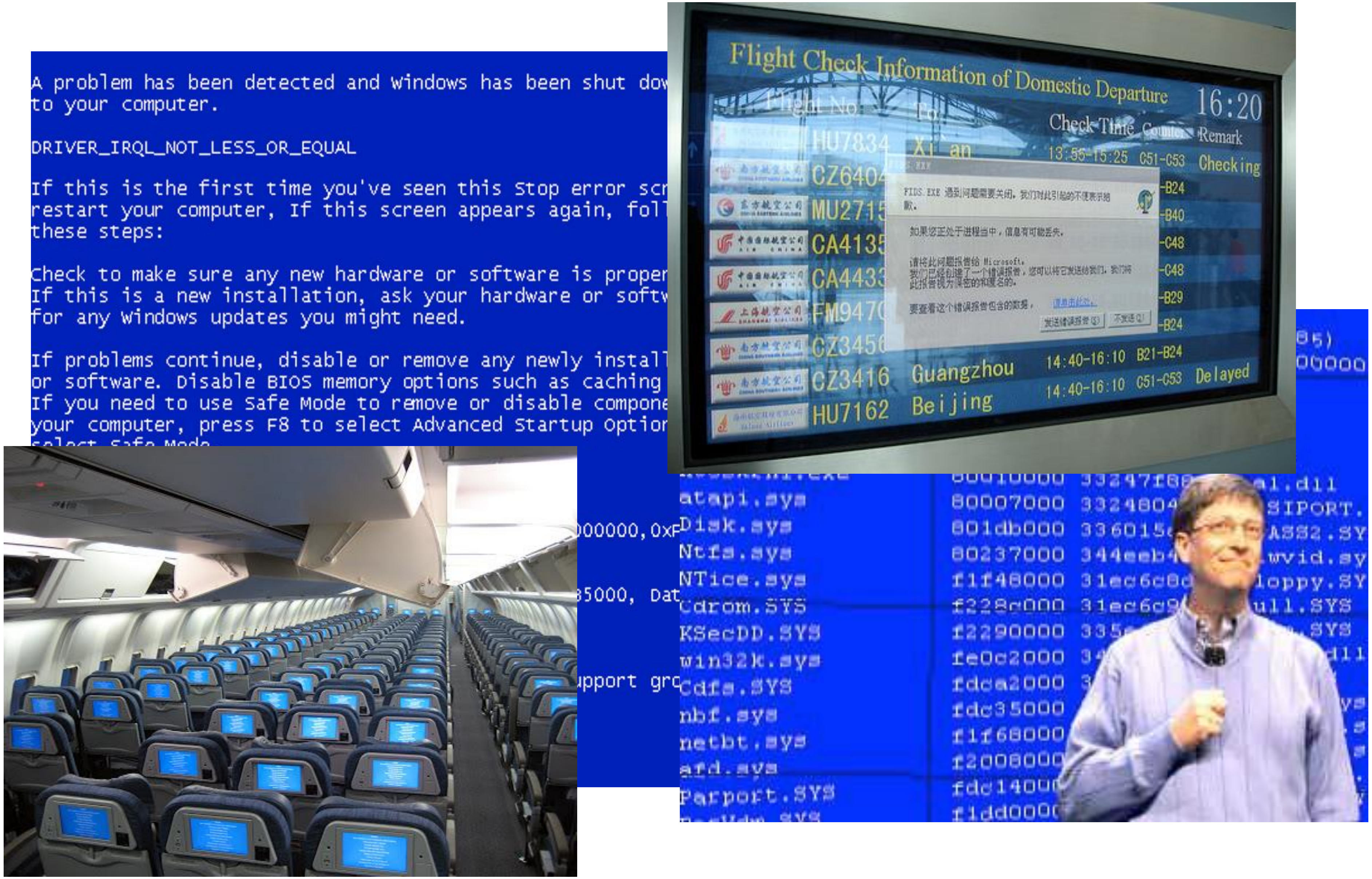
Errors

Facts About Computer Errors

- Error-free software is not possible.
- Errors are often caused by more than one factor.
- Errors can be reduced by following good procedures and professional practices.

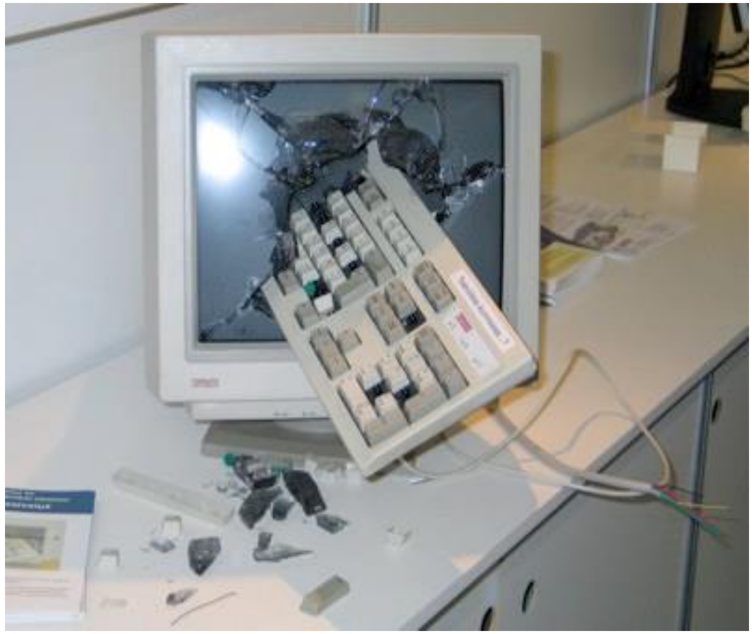
Can we trust computers?

Software failures



Hardware Failures

Failures due to external causes



Fire in TIM's server farm (Naples)

BOEING 737 MAX

Some basic terminology

- From the Webster's Definitions….

- Dependable: *capable of being depended on: RELIABLE*

  (to trust someone or something and know that they will help you or do what you want or expect them to do. The dependability of a system reflects the user's degree of trust in that system)

- Reliable: *suitable or fit to be relied on: DEPENDABLE*

Some basic terminology

- Rely:
  - To be dependent <*the system for which we depend…*>
  - To have confidence based on experience <*someone you can rely on*>

Dependability

- “...the ability to deliver service that can justifiably be trusted” (Laprie, 2001)

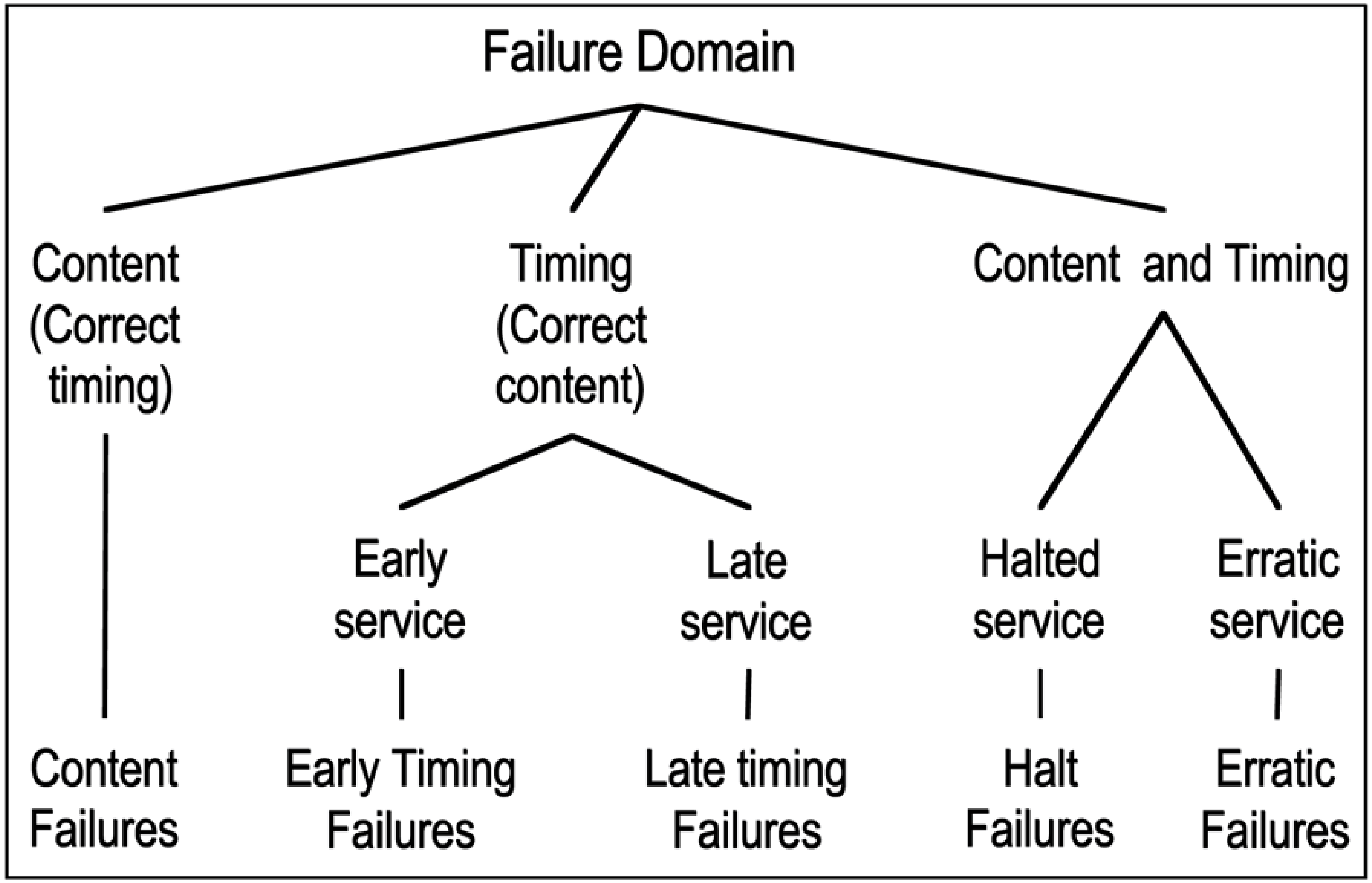
- “...ability to avoid service failures that are more frequent and more severe than acceptable”* (Laprie, 2004)

**Acceptability <u>is subject to a system’s application context</u>*

So, dependability may change if the system is applied in different contexts, by different users, or if the application context evolves over time.

The Dependability Tree

The role of time in the failure domain



Safety

- The safety is the probability of absence of **catastrophic failures**, i.e. failures that lead to catastrophic consequences on the users or on the environment

$$S(t)=P(!CatastrophicFailure\ in\ (0,t))$$

- A failure is "declared" *catastrophic* on the basis of risk analysis procedures

- **Fail-safe system**: A system whose failures are, to an acceptable extent, all minor ones (non catastrophic)

Safety

- Are commercial aircrafts safe?
  - They crash very occasionally
- How many crashes are too many?
- Are cars “safe”?
  - They crash quite a lot

45K deaths/year;

900/week = 2 fully loaded Boeing 747/week

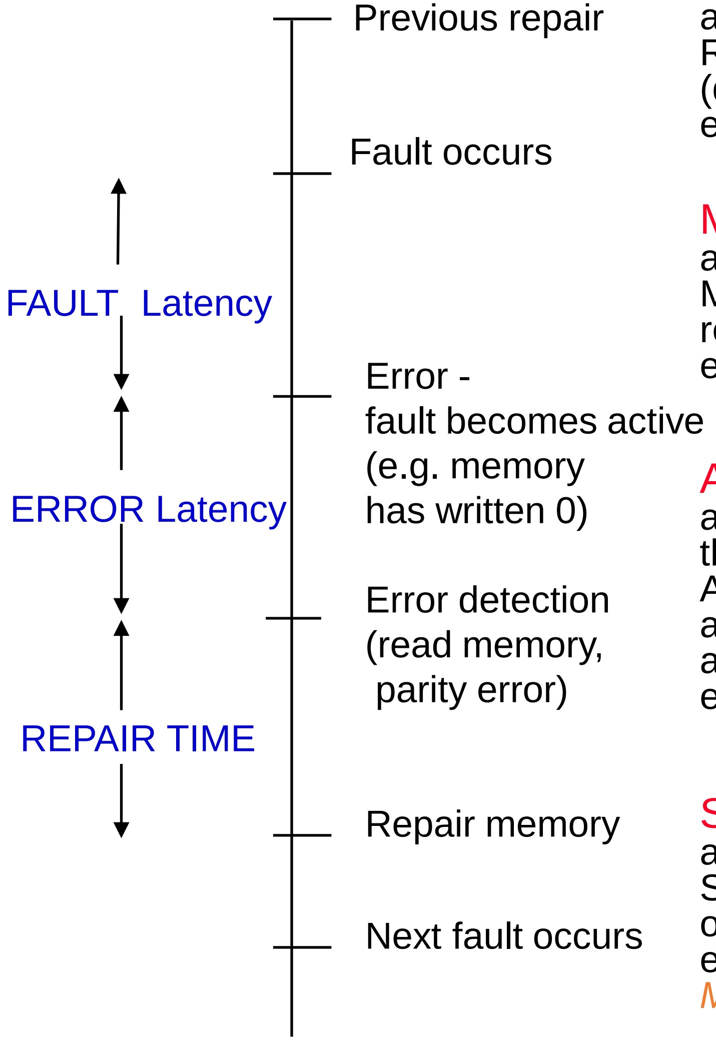
Risk

Risk is the expected loss per unit time

$$Risk = \Sigma\ pr(accident_i) \times cost(accident_i)$$

Safety is expressed as an acceptable level of loss

Overall Picture



Reliability:
a measure of the continuous delivery of service;
R(t) is the probability that the system survives (does not fail) throughout [0, t];
expected value: *MTTF(Mean Time To Failure)*

Maintainability:
a measure of the service interruption
M(t) is the probability that the system will be repaired within a time less than t;
expected value: *MTTR (Mean Time To Repair)*

Availability:
a measure of the service delivery with respect to the alternation of the delivery and interruptions
A(t) is the probability that the system delivers a proper (conforming to specification)service at a given time t.
expected value: *EA = MTTF / (MTTF + MTTR)*

Safety:
a measure of the time to catastrophic failure
S(t) is the probability that no catastrophic failures occur during [0, t];
expected value:
MTTCF(Mean Time To Catastrophic Failure)

Safety vs. Reliability vs. Availability

- They are not the same…..
- Example
  - A system that is turned off every day
    - *Is not very reliable*
    - *Is not very available*
    - *But it is probably very safe*
- *When the airline says: "Safety is our highest priority"….. ☺*
- In practice, safety often involves specific intervention

Dependable Computing

- **Dependable computing** is the correct execution of a specified function in the presence of faults.
- Techniques to increase dependability can be divided into two basic approaches:
  - **Fault intolerance** (fault avoidance)
  - **Fault tolerance**

Real-time and safety

- Realize systems that meet all temporal constraints *by design*
  - *Fault avoidance*
- *Or*
- Realize systems able to react to failures in a bounded time
  - *Fault tolerance*

Safety Standards

- Standards for the certification of products and processes by third party validators, to reduce the potential for dangerous malfunction resulting from people, environment and/or process
- The SIL concept (Safety Integrity Level) in IEC 61508
  - A SIL is defined as 'a discrete level (one of 4) for specifying the safety integrity requirements of safety functions'. Thus, a SIL is a target probability of catastrophic failure of a defined safety function
  - SIL1 has the lowest level of risk reduction. SIL4 has the highest level of risk reduction.

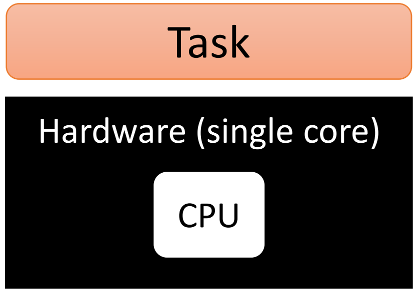
Design and development issues

- Often, real-time systems are designed and developed using empirical techniques
  - Management of temporal events trough fat portions of assembler code
  - Development of control actions in low-level drivers for the management of I/O devices
  - Direct management of interrupt priorities
- Although the produced code could result efficient and (perhaps) certifiable, it entails several problems:
  - Laborious programming
  - Difficult understanding and maintenaibility of code
  - Difficult verification of temporal constraints

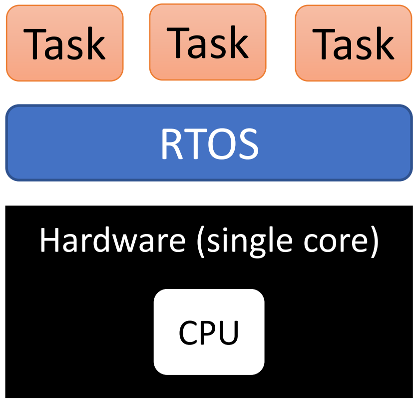
The role of real-time operating systems

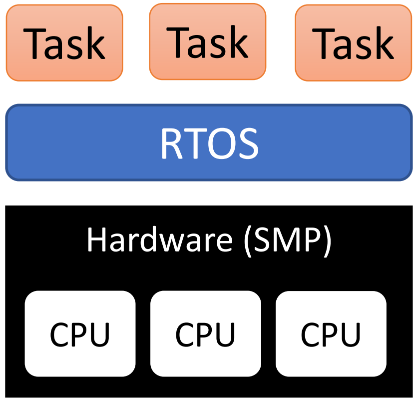
- Recent real-time systems are increasingly using real-time operating systems (**RTOS**), which introduce core mechanisms that are thought to explicitly manage the time variable, such as:
  - Job scheduling
  - Mutual exclusion and synchronization
  - Message-exchange communication
  - Managing outages and memory
  - ...
- This allows to build control software
  - with high-level programming languages
  - more robust, in terms of time constraints satisfaction
- But, in safety-critical systems the RTOS needs to be certified

(software) Evolution pathways of real-time industrial systems

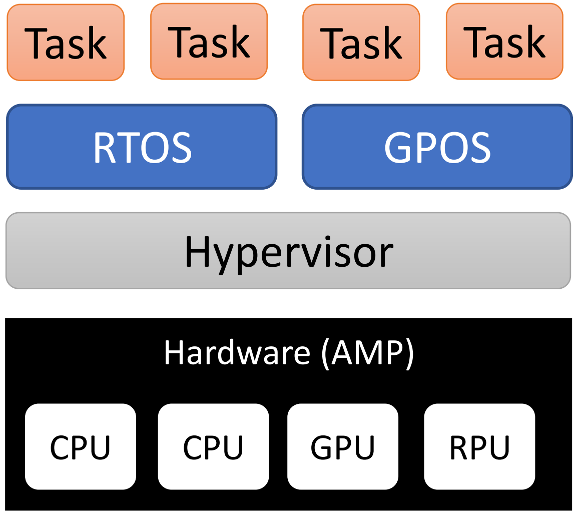


- *Bare metal*
- *Deeply embedded*
- *Single purpose*





- *Operating system*
- *Multi-tasking*
- *Single-purpose*



- *Hypervisor*
- *Virtual machines*
- *or containers*
- *Multi-purpose*

(software) Evolution pathways of real-time industrial systems

- From bare metal to OS abstraction
- From single core to multi-core and asymmetric multi processing
- From single OS to multiple virtual envs (virtualization)
- From single purpose to multi purpose
  - ... so from single criticality to mixed criticality

(software) Evolution pathways of real-time industrial systems

- Many challenges ahead!
  - How to schedule real-time tasks on multi-cores?
  - How to avoid/reduce non-determinism?
  - How to provide isolation to virtual machines and/or containers?
  - How to manage heterogenous processors?

Course roadmap

- Recap on real-time systems
  - Scheduling of real-time tasks, Resource management, Aperiodic servers
- Linux real-time extensions
- Programming real-time tasks in Linux
- Real-time monitoring
- Mixed-criticality systems
  - System models, hierarchical scheduling
- Multiprocessor real-time scheduling
- Real-time virtualization
  - Over SMP and AMP
- Industry standards (safety and platforms)
  - Automotive, Avionics, Railways, Industry 4.0 and IIoT

Industrial Internet of Things Architecture