Real-Time Systems
Achievements and Perspectives

Giorgio Buttazzo

Retis
Real-Time Systems Laboratory

Keynote at the 35th IEEE Real-Time Systems Symposium, Rome, 2014

Retis Real-Time Systems Laboratory

Outline

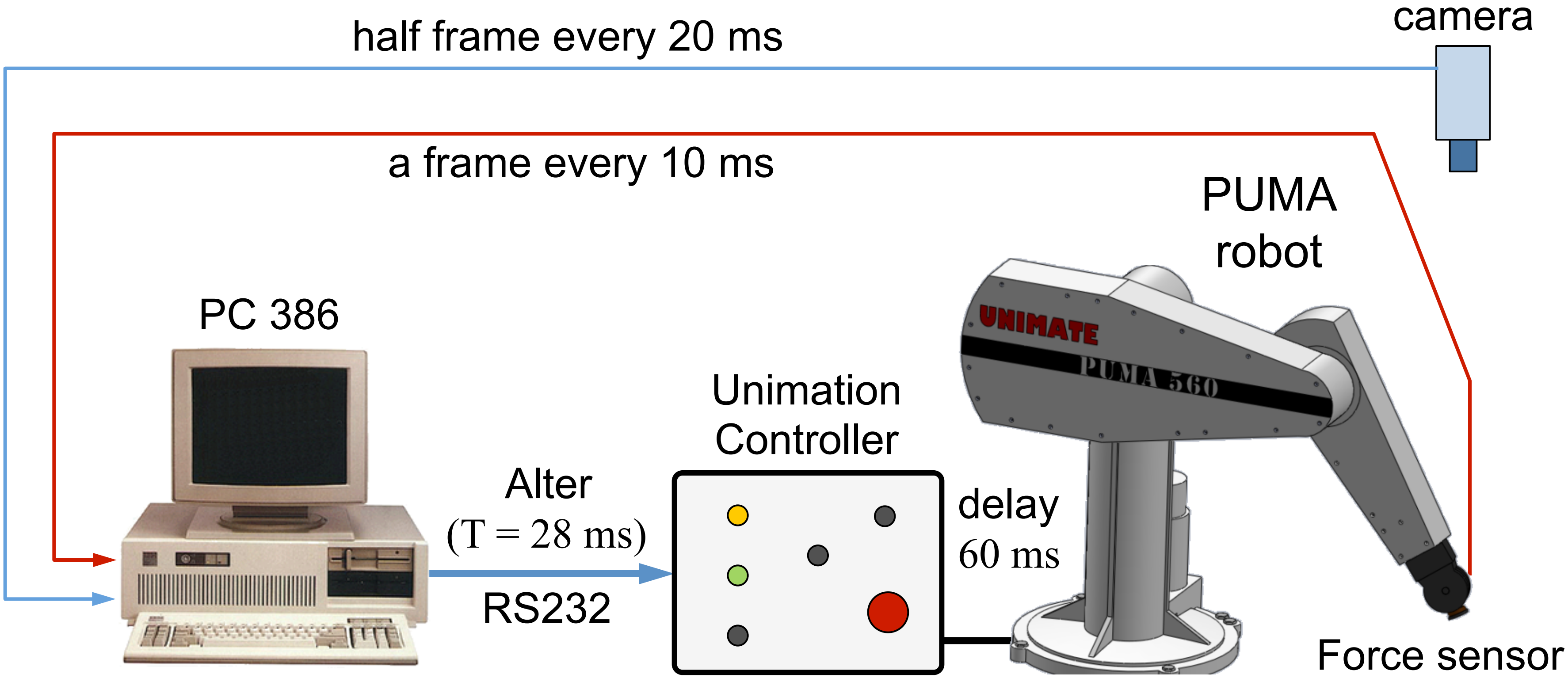
1. A quick look into my past

2. Major achievements in the RT community

3. A look into the future

Back to 1987

- I started my PhD working on robotics
- Active perception and sensory-motor coordination



Need for RT support

Very soon, I realized a strong need for RT support:

- Scheduling: periodic & aperiodic tasks
- Mixed criticality: hard & soft deadlines
- Data sharing: async. comm. among periodic tasks
- Analysis: effects of delay and jitter on performance

Crucial references

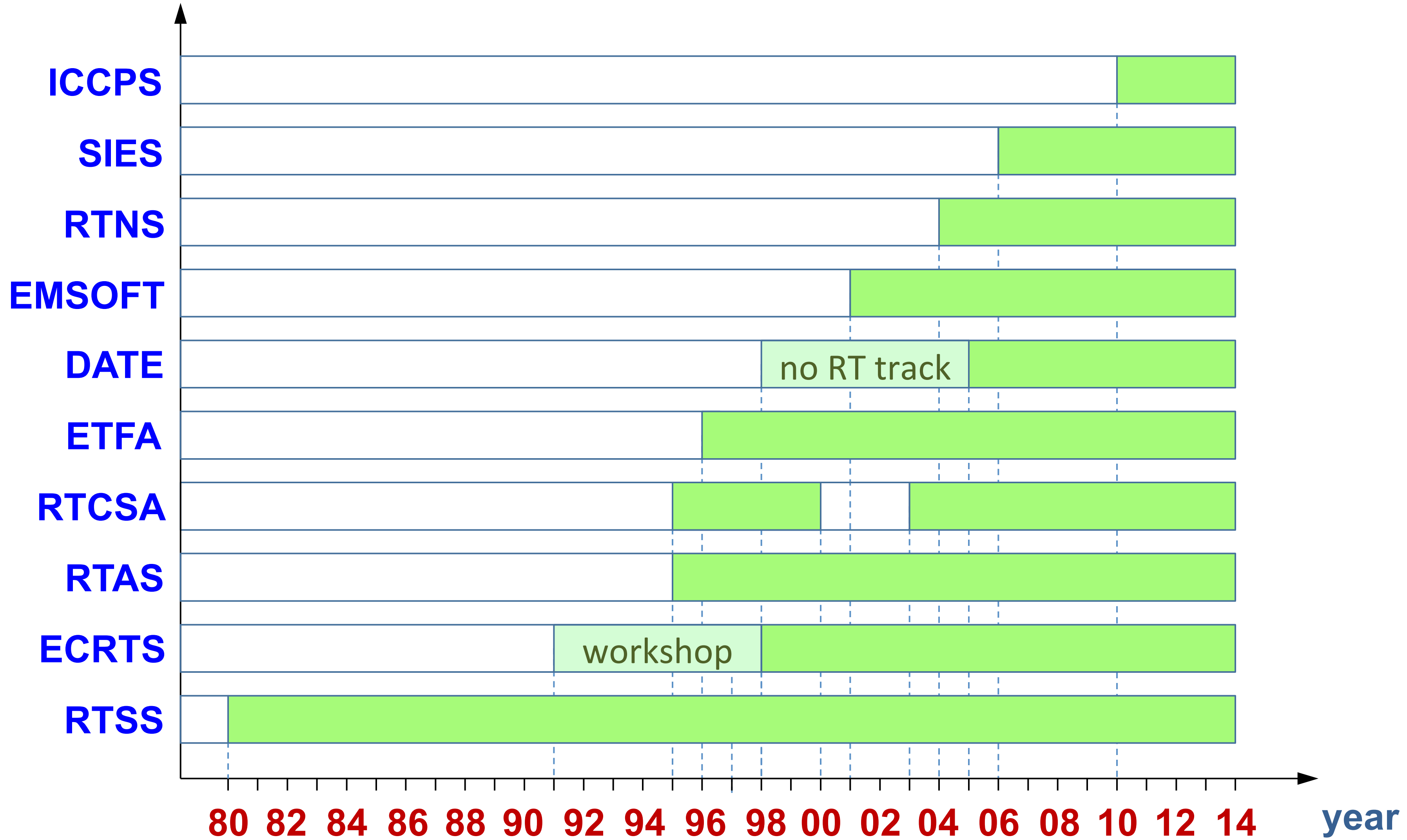
- J. Stankovic, "Misconceptions about real-time computing: a serious problem for next-generation systems", IEEE Computer, 21 (10), 1988.
- J. Stankovic & K. Ramamritham, Hard Real-Time Systems: Tutorial, 1988. (*collection of 47 papers on RT computing*)
- Special Issue on Real-Time Kernels, ACM Operating Sys. Review, 1989.

The change: 1993

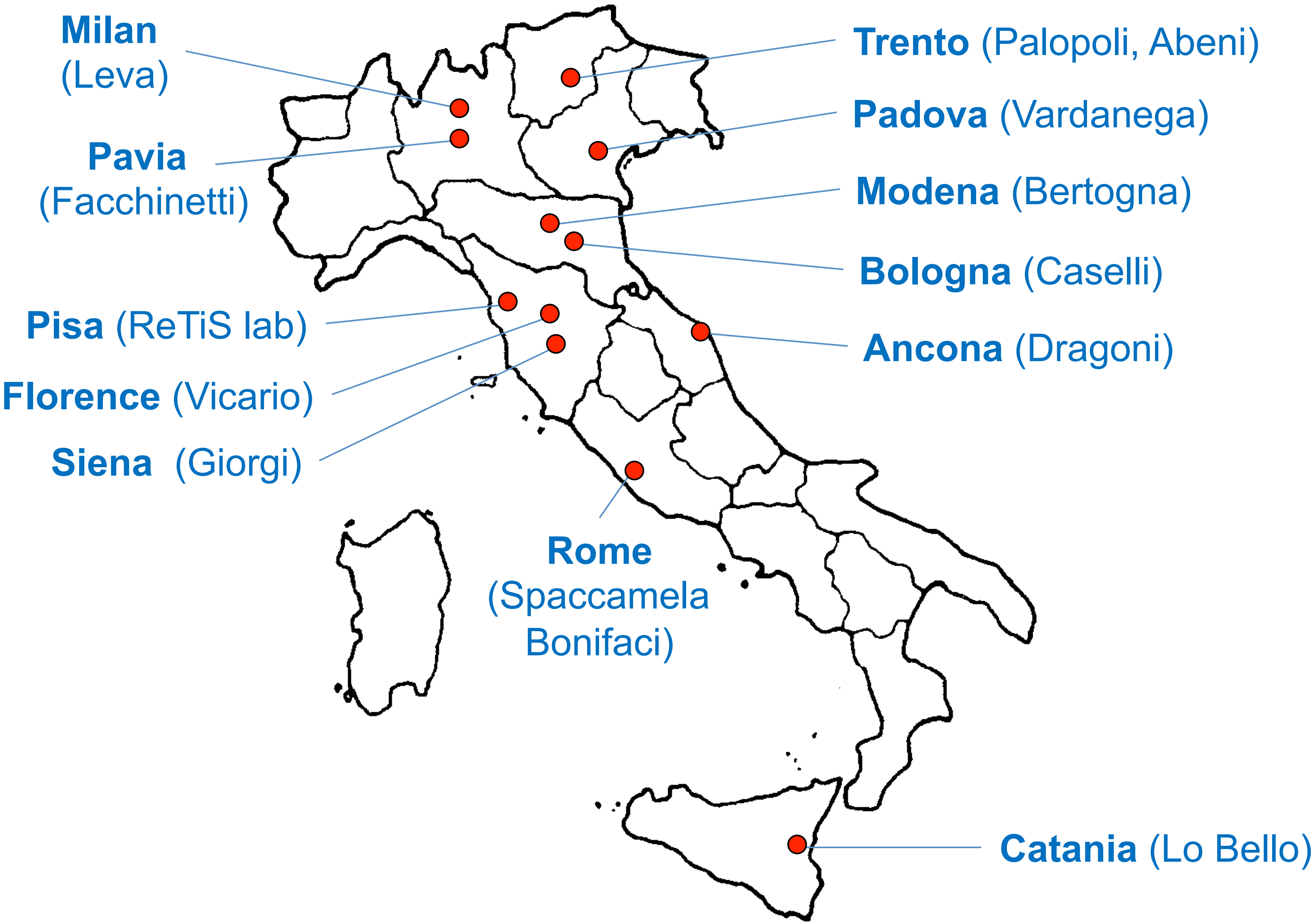
- I heard Jack Stankovic was looking around for sabbatical, so I invited him in Pisa.
- At that time there were not PDF files, so he brought a lot of RT papers to read ... very useful … thanks Jack!

RTSS 1993
(Raleigh-Durham, NC, USA)
was my first RT conference

RT Conferences



Groups on RTS in Italy



Groups on RTS in Europe

Austria: Vienna

Czech Rep: Prague

France: Paris, Grenoble, Renne, Nantes, Nancy, Toulouse, Lille

Germany: Munich, Kaiserslautern, Dresden, Karlsrue, Saarland

Ireland: Dublin, Cork

Italy: Pisa, Pavia, Catania, Siena, Florence, Bologna, Trento, Padova, Modena, Ancona, Rome

Portugal: Porto, Aveiro, Lisbon

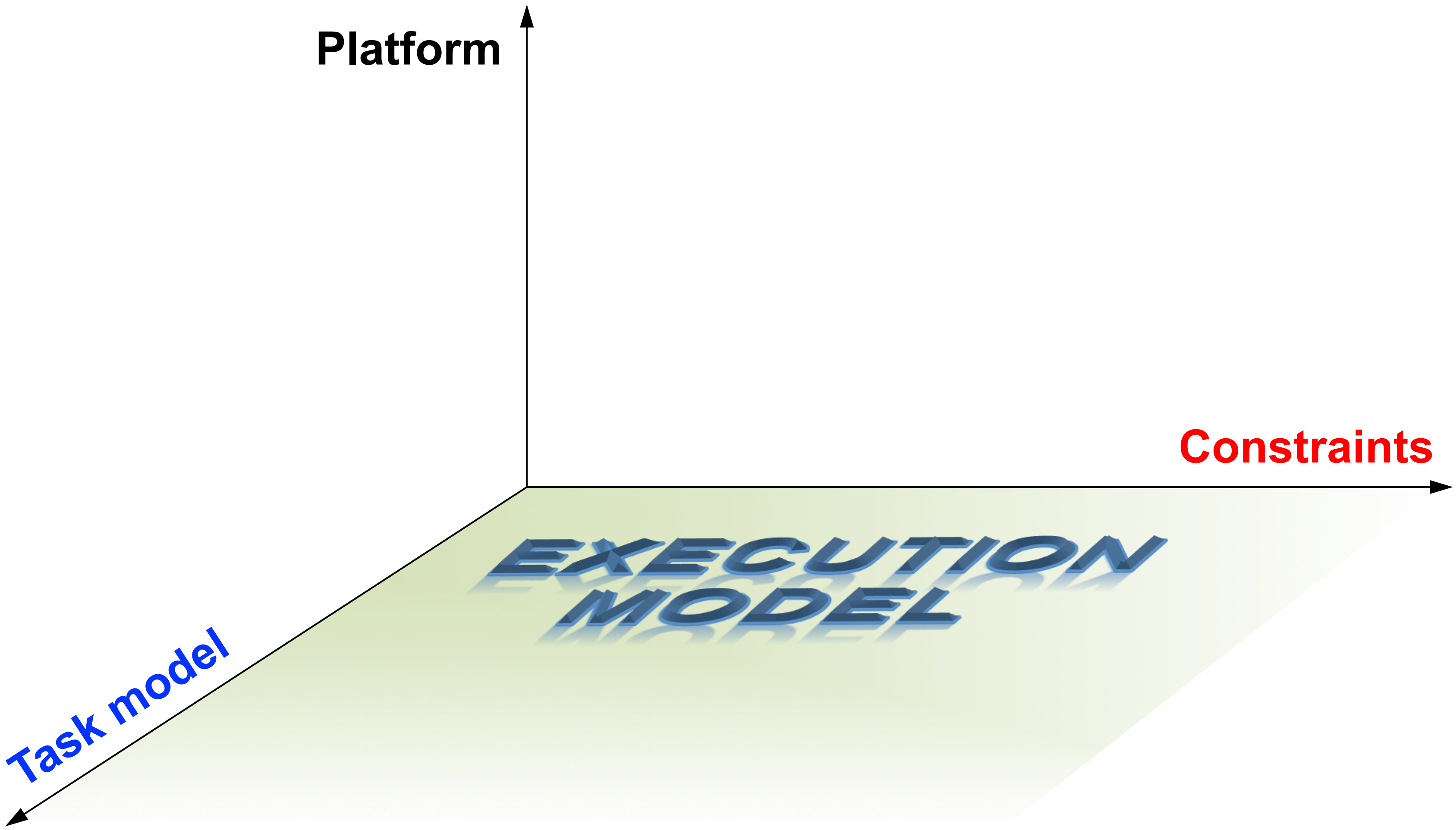
Spain: Madrid, Cantabria,Valencia, Barcelona,, Palma de Mallorca

Sweden: Stockholm, Lund, Vasteras, Uppsala, Halmstad, Linkoping

Switzerland: Zurich, Lausanne,

UK: York

Solved problems



Execution Model

Task models

Non recurring (single jobs)

Recurring

- Aperiodic
- Sporadic
- Periodic
- Strictly periodic
- Multiframe
- DAGs
- Digraf
  - Skip model (m,k)
  - Mixed criticality
  - Elastic
  - Imprecise computation
  - Variable Rate

Constraints

- Precedence
- Resources
- Self-suspensions
- Mode changes
- Fully Preemptive
- Fully Non preemptive
- Limited preemptive
  - Preemption thresholds
    Floating regions
  - Deferred preemptions
  - Fixed preemption points

Real-Time Systems Laboratory

Platforms

Uniprocessor

- Fixed speed
- Variable speed (DVFS)
  - continuous
  - discrete
- Low power states (DMP)
- With cache

Multiprocessor

- Identical
- Uniform
- Heterogeneous
- Hybrid

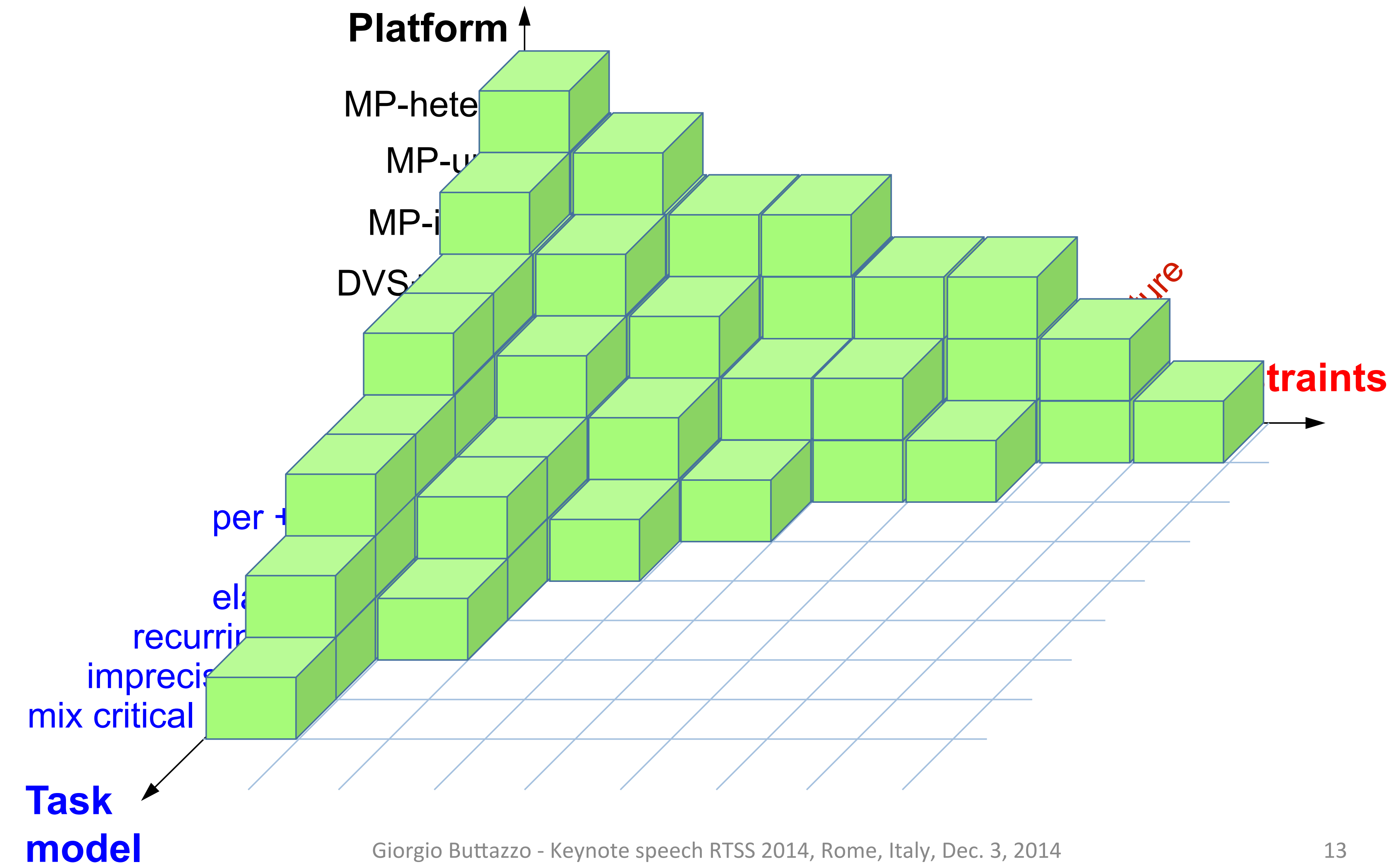
Distributed

- Wired
  - crossbar
  - mesh
  - star
  - tree
  - bus
- Wireless
  - single-hop
  - multi-hop

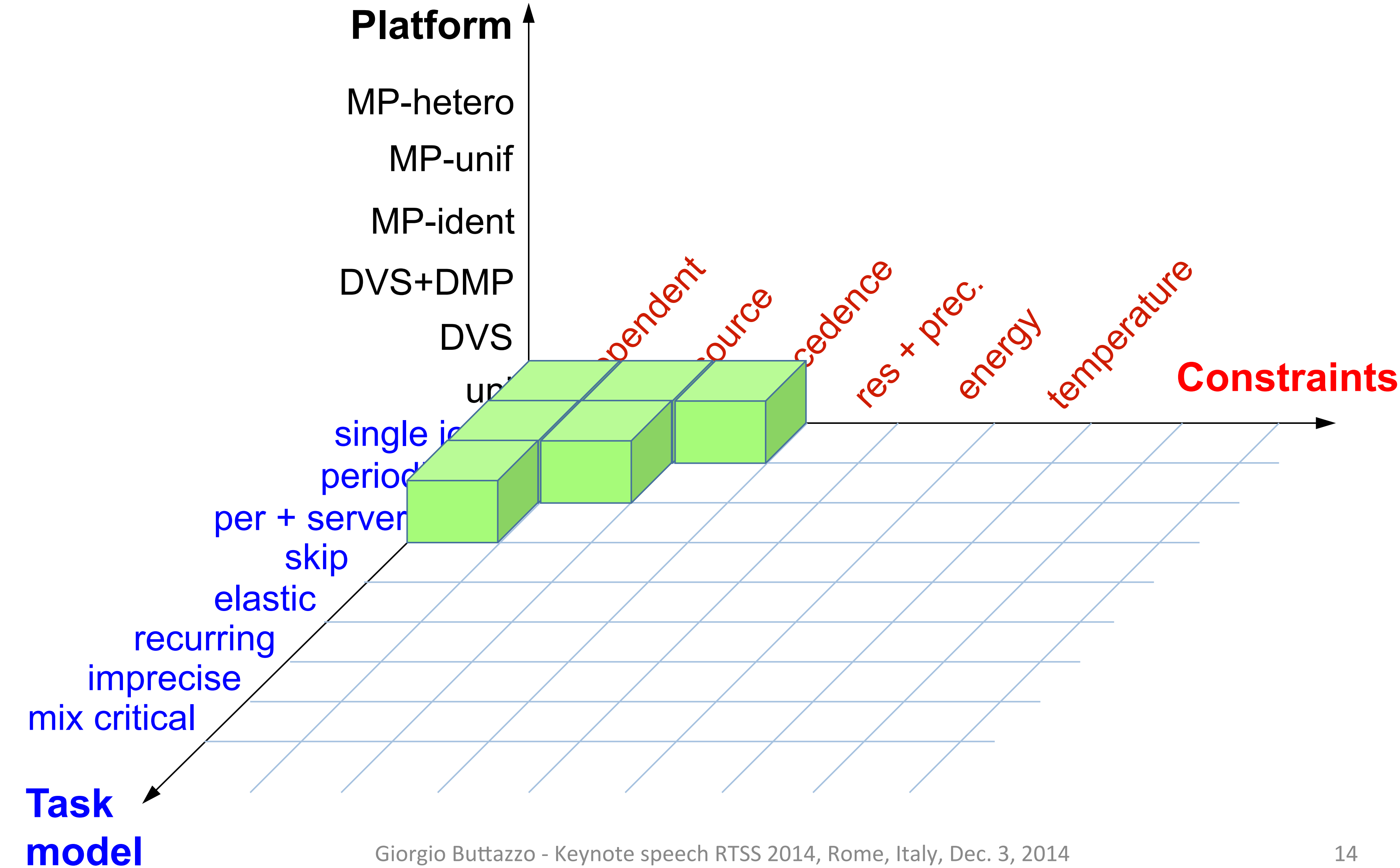
Optimization criteria

- Feasibility
- Response time
- Maximum lateness
- Utilization bandwidth
- Energy consumption
- Temperature
- Number of processors

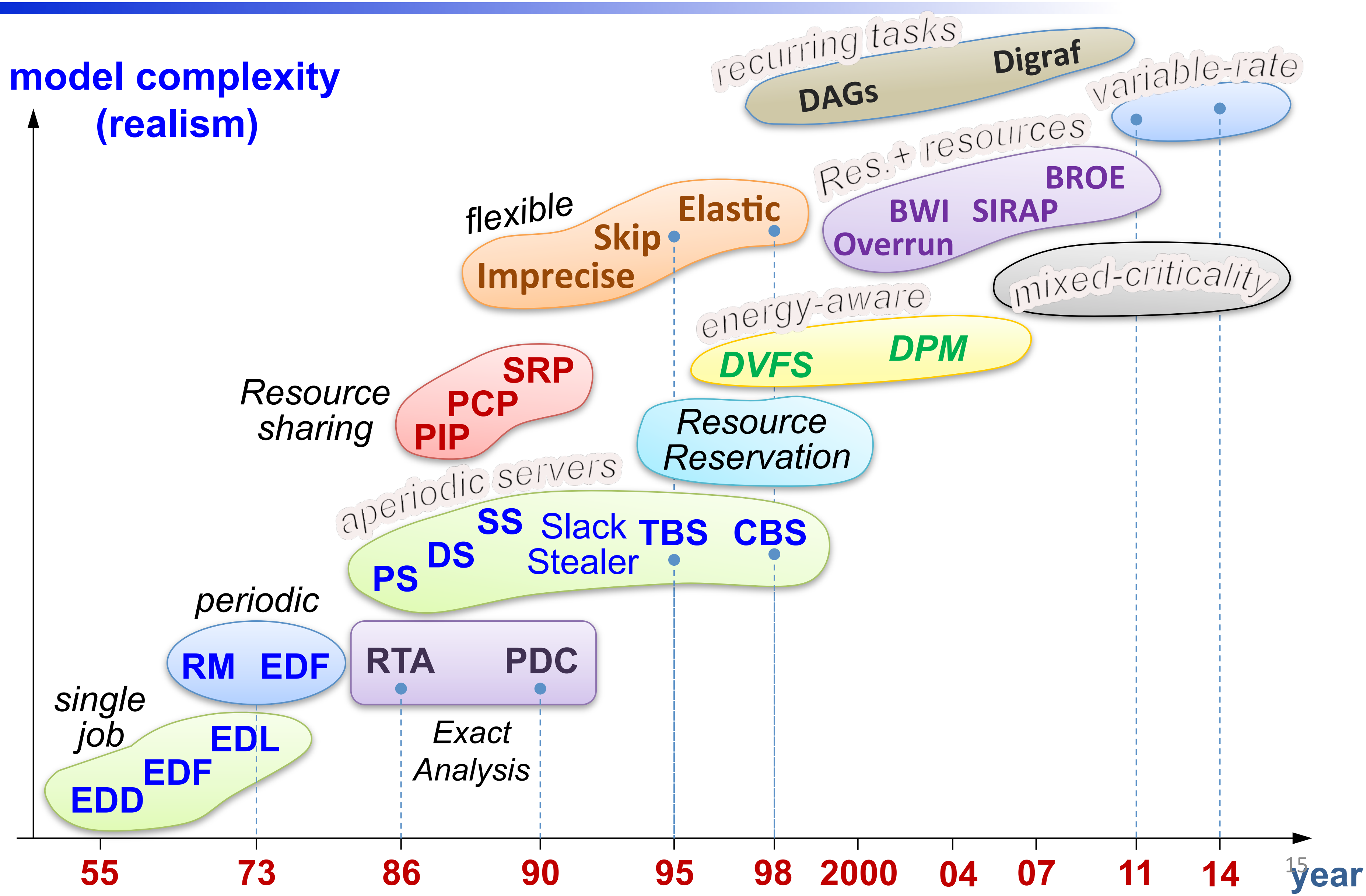
Results on RTS



What is really being used?



Some major achievements



Concrete impact

- **Rate Monotonic** is used in most industrial settings
- **Priority Inheritance** is in most RT kernels
- **Sporadic Server** is specified in POSIX
- **CBS** is implemented in LINUX
- **EDF** is now supported by a few kernels
  - **Erika Enterprise** (by Evidence): certified **OSEK** and adopted by Magneti Marelli in next generation ECUs
  - **Ada 2005** runtime support
  - **Linux** (**SCHED_DEADLINE** in mainline since June 2014)
- Lots of **RT tools** are now available for
  - WCET estimation, schedulability analysis, scheduling simulation, formal verification, etc.

From RTS to CPS

Are they just buzzwords?

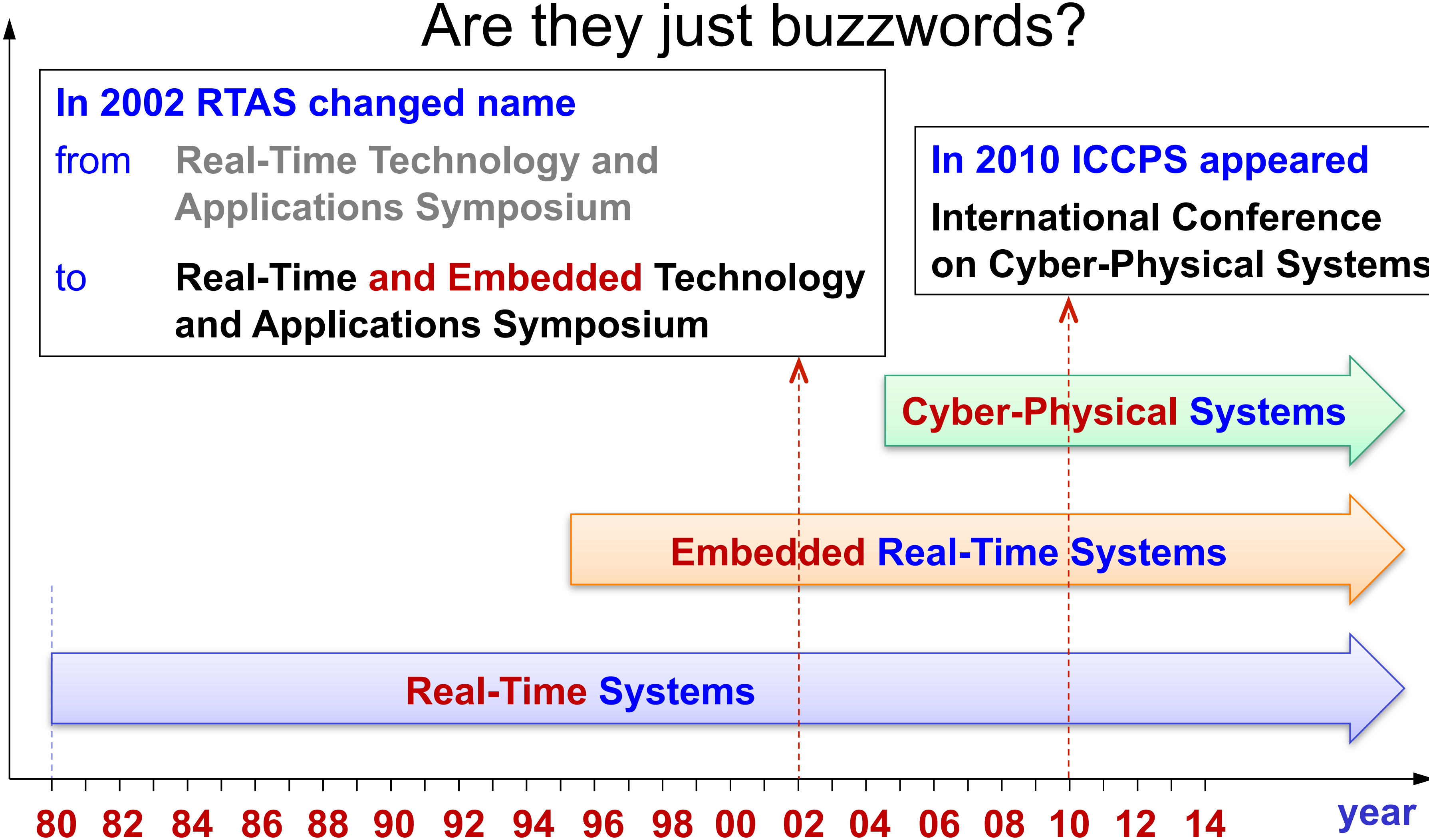
In 2002 RTAS changed name

from Real-Time Technology and Applications Symposium

to **Real-Time and Embedded Technology and Applications Symposium**

In 2010 ICCPS appeared

International Conference on Cyber-Physical Systems



Retis
Real-Time Systems Laboratory

Definitions

Real-Time System

Computing system able to provide bounded response times to tasks with bounded execution, in all possible scenarios.

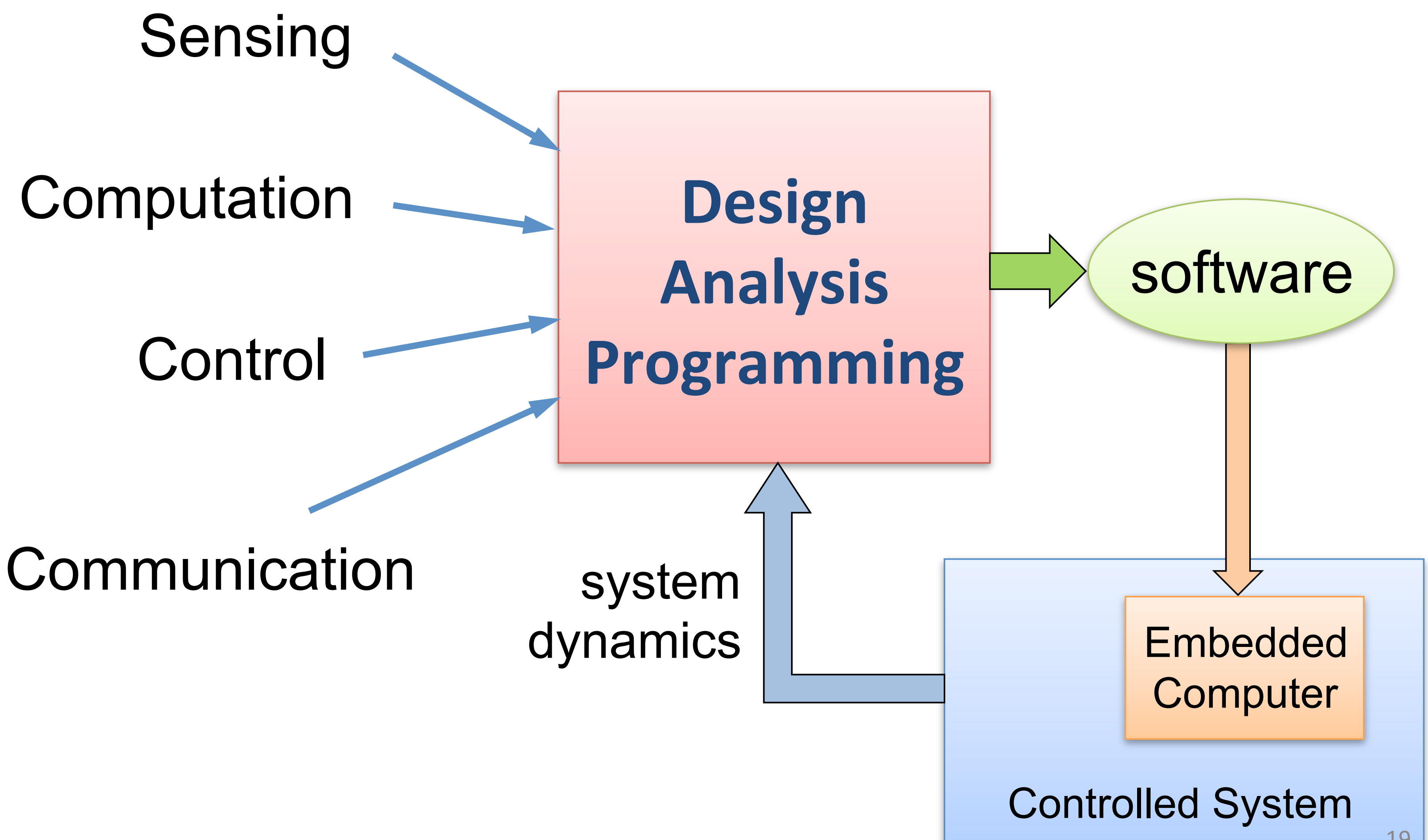
Embedded System

Computing system embedded into a larger device, dedicated to control its functions, manage the available resources, and simplify the interaction with the user.

Cyber-Physical System

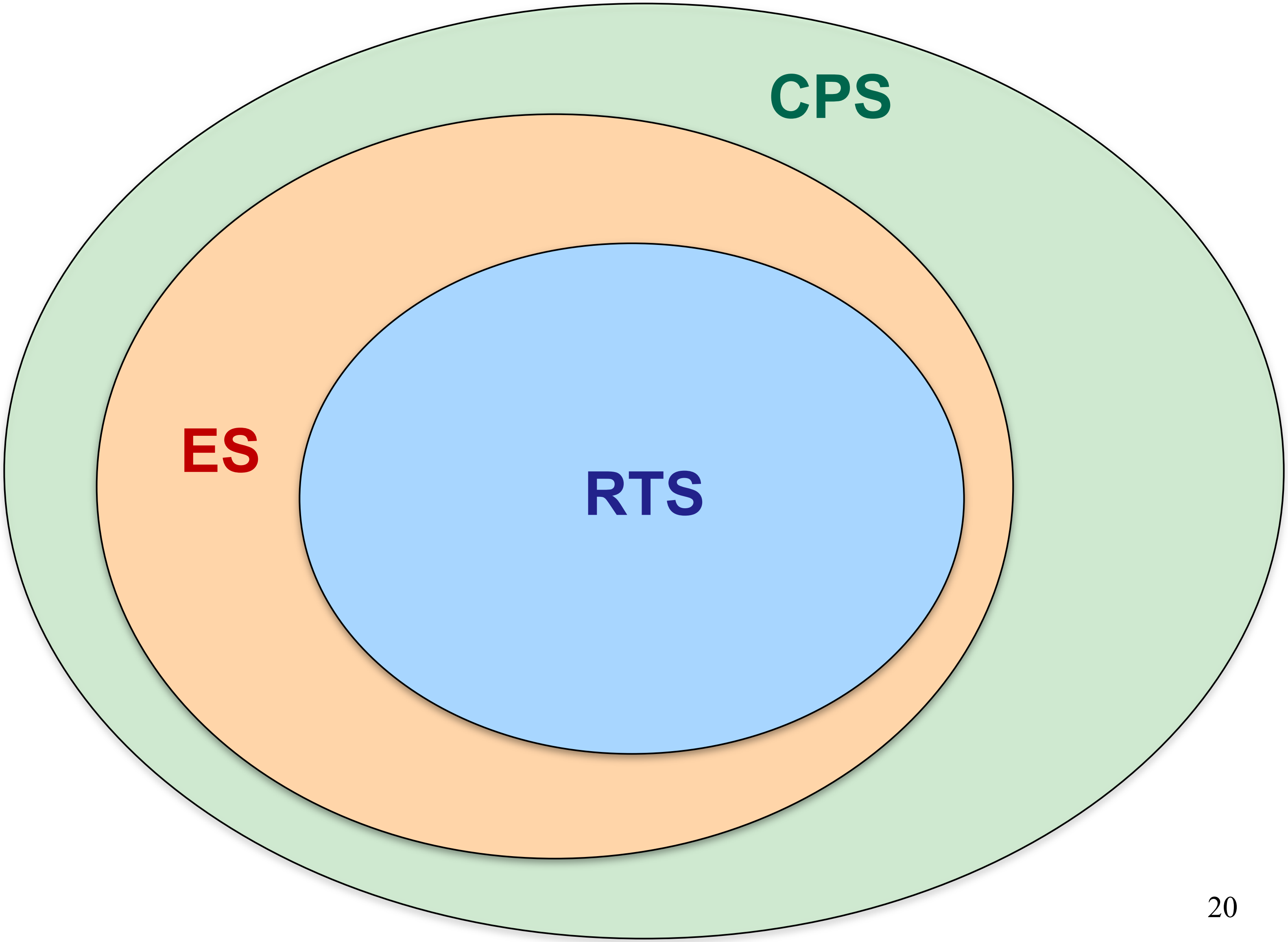
System where software is tightly coupled with the physical characteristics of the plant to be controlled.

Cyber-Physical Systems

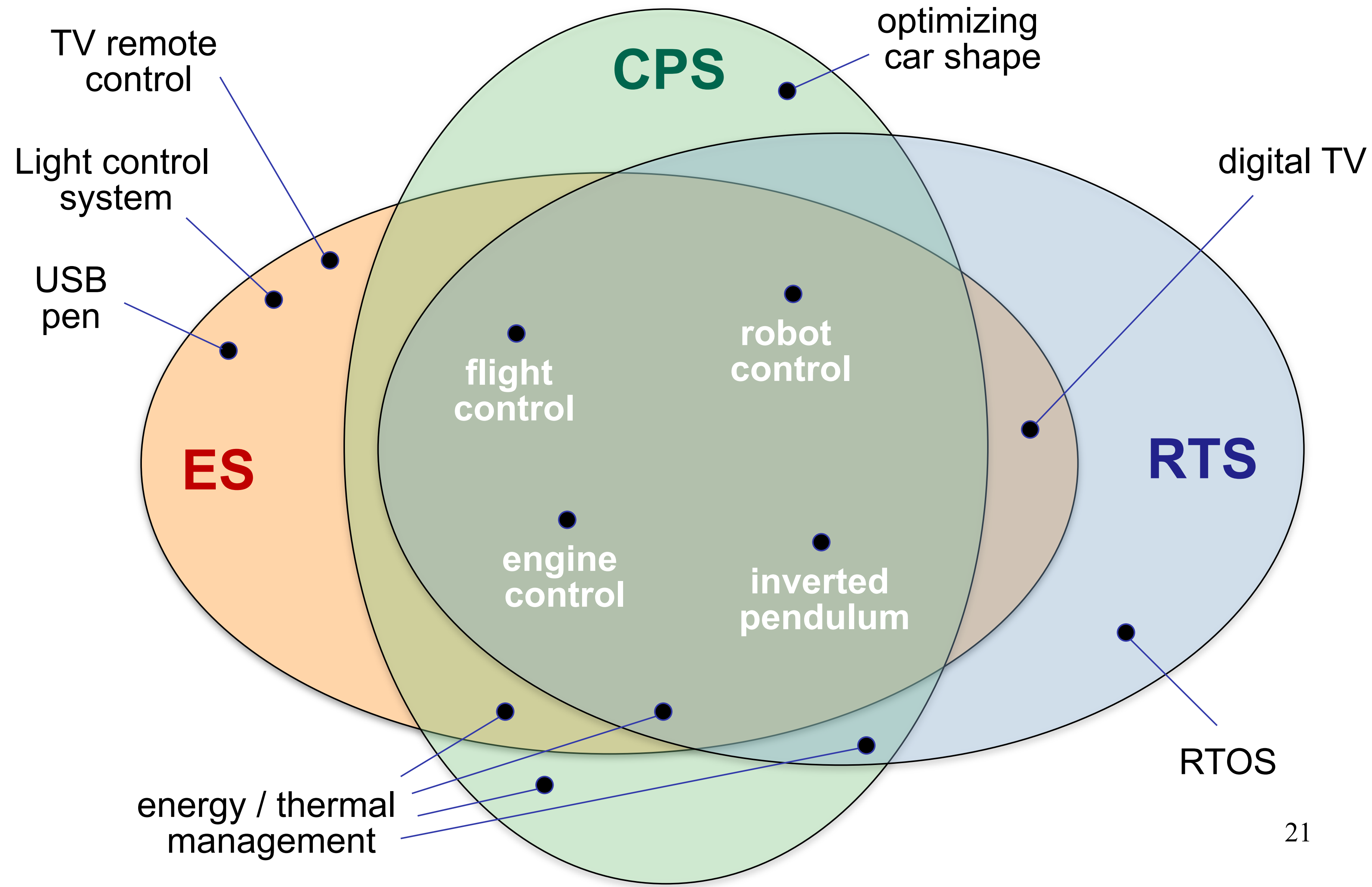


Retis
Real-Time Systems Laboratory

One (wrong) view



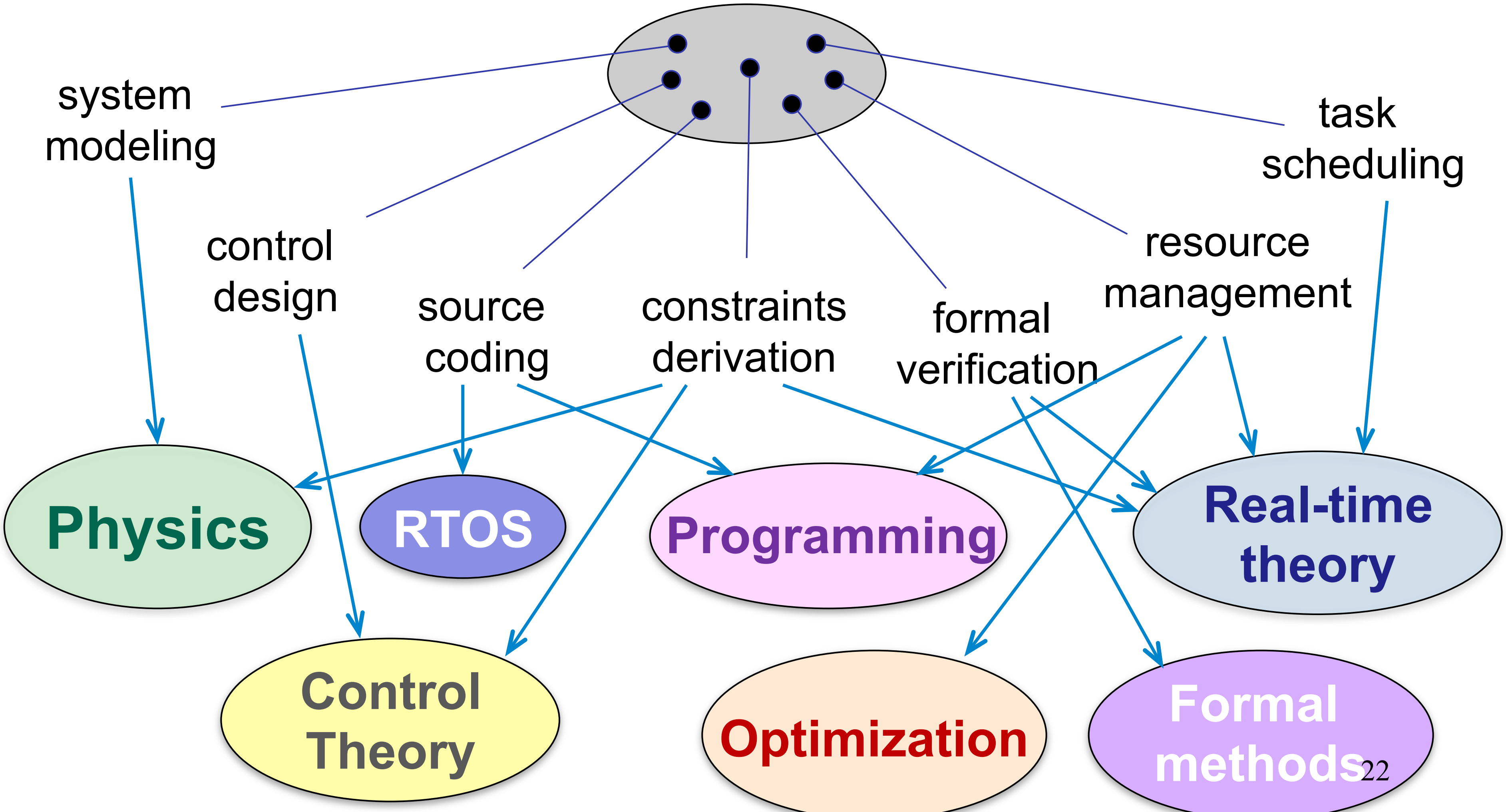
My view



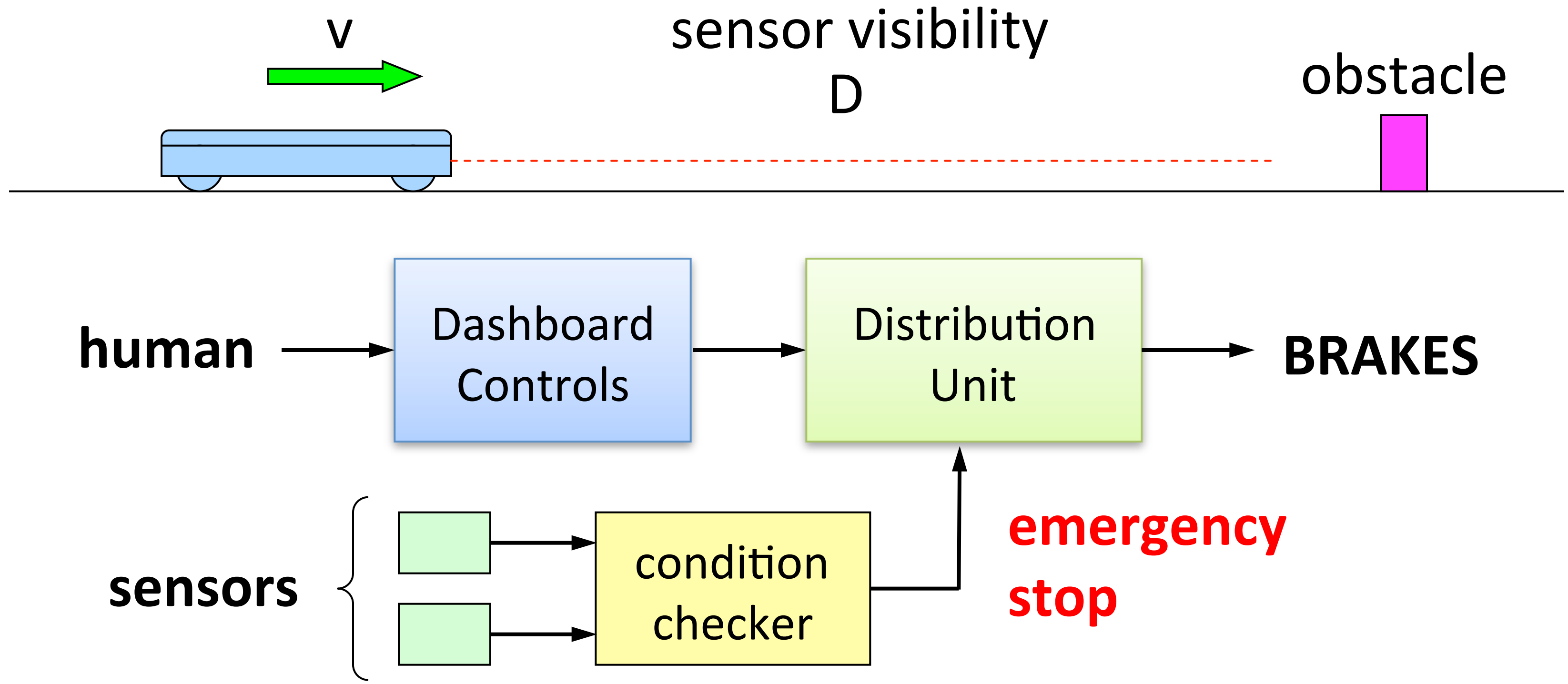
CPS
ES
RTS
TV remote control
Light control system
USB pen
optimizing car shape
digital TV
flight control
robot control
engine control
inverted pendulum
RTOS
energy / thermal management

The truth is that…

Complex systems require interdisciplinary knowledge



An instructive example

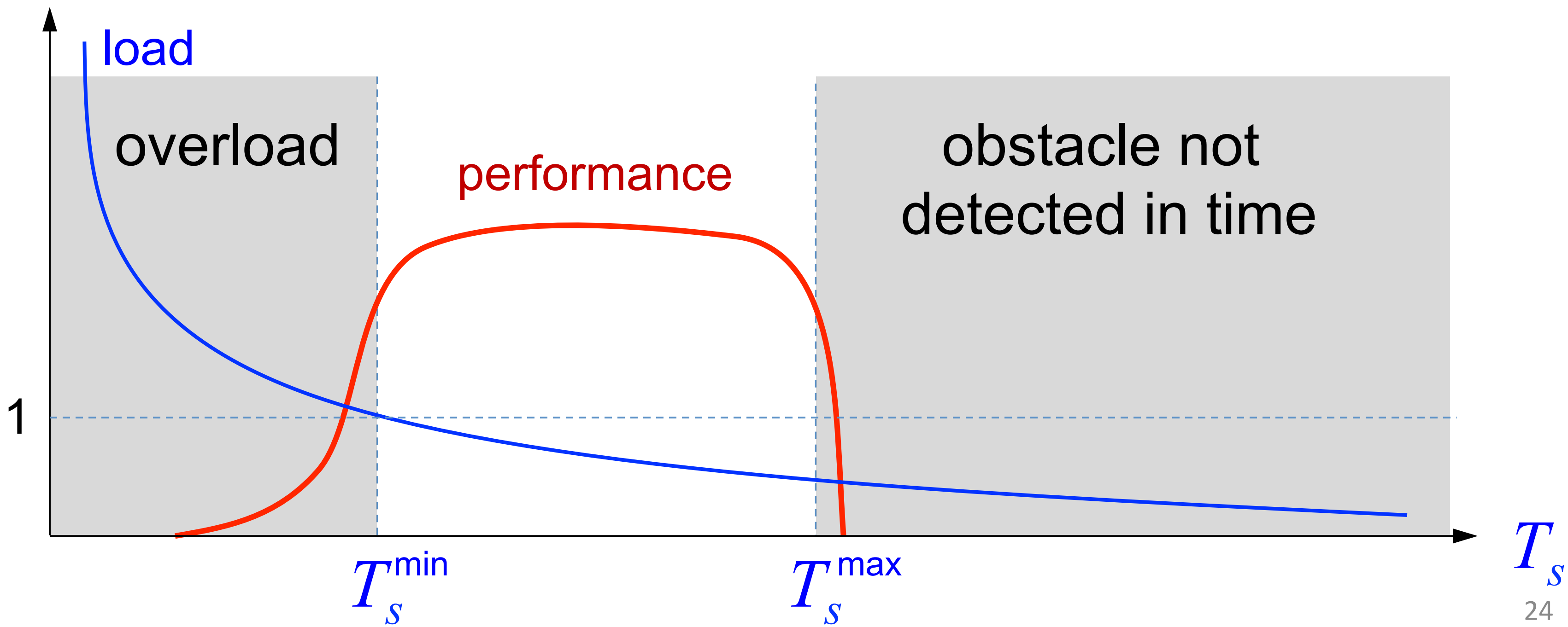


GOAL: If an obstacle is detected, stop the train without hitting the obstacle.

PROBLEM: Find the sampling periods of the sensors that guarantee the feasibility of the goal

Assumptions

- Tasks are scheduled by Rate Monotonic (implicit deadlines)
- Let $\tau_s(C_s, T_s)$ be the periodic task devoted to sampling
- Assume τ_s has the shortest period (highest priority).
- Let U_{other} be the load of the other tasks



Minimum period

The minimum period can be computed imposing the system schedulability by the Liu & Layland test:

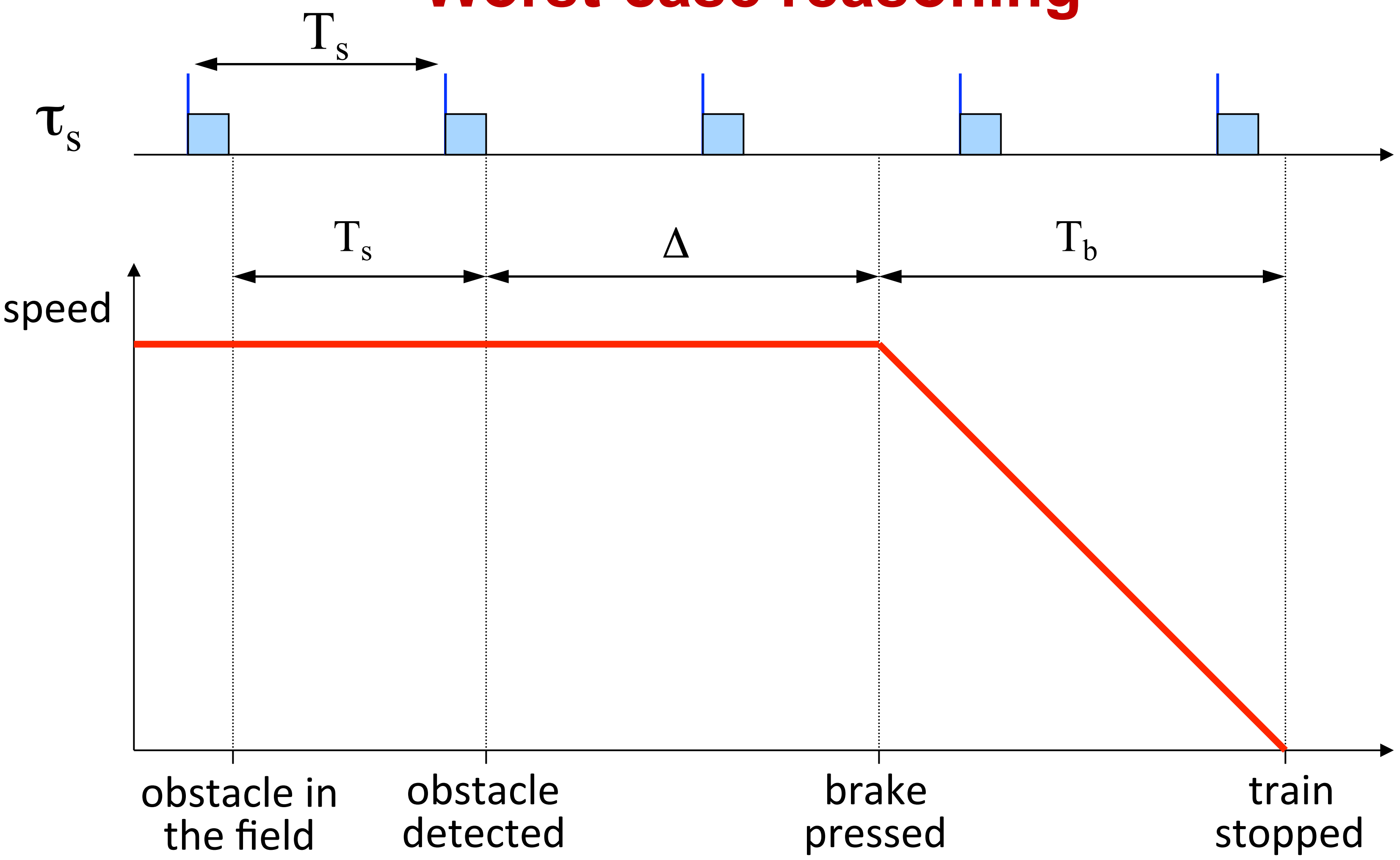
The system is schedulable if $$\frac{C_s}{T_s} + U_{other} \le U_{\text{lub}}^{RM}$$

that is $$T_s \ge \frac{C_s}{U_{\text{lub}}^{RM} - U_{other}}$$

$$\boxed{T_s^{\min} = \frac{C_s}{U_{\text{lub}}^{RM} - U_{other}}}$$

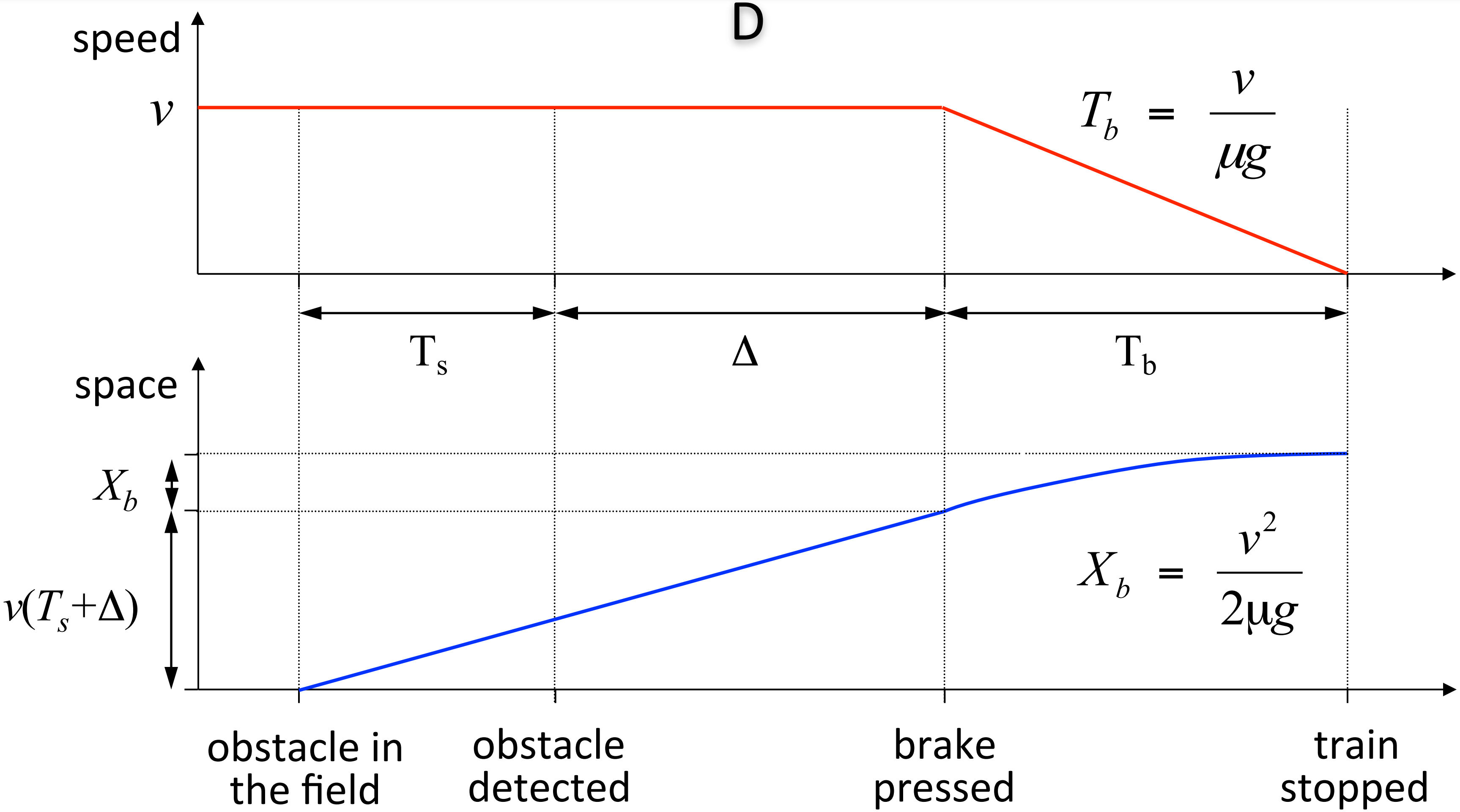
Maximum period

Worst-case reasoning



Safety condition

The space covered by the train in $(T_s + \Delta + T_b)$ should not exceed D



Safety condition

$$v(T_s + \Delta) + X_b < D$$

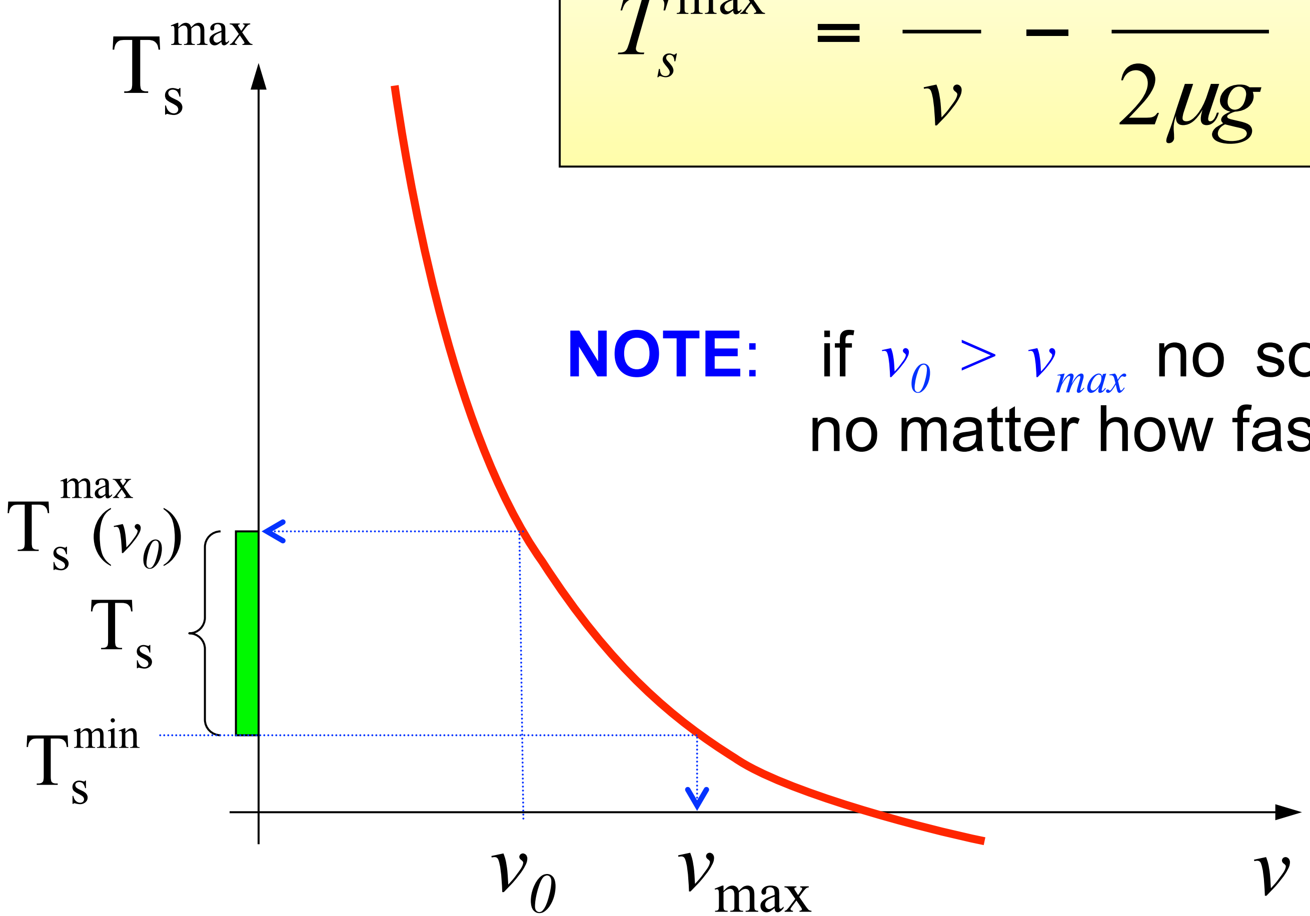
$$X_b = \frac{v^2}{2\mu g}$$

$$v(T_s + \Delta) + \frac{v^2}{2\mu g} < D$$

$$T_s < \frac{D}{v} - \frac{v}{2\mu g} - \Delta \longrightarrow T_s^{\max}$$

Safety condition

$$T_s^{\max} = \frac{D}{v} - \frac{v}{2\mu g} - \Delta$$



NOTE: if $v_0 > v_{max}$ no solution exists, no matter how fast the CPU is!

How can we increase impact?

Producing and publishing results is not enough…

- Show concrete benefits

- Convince industrial people (not easy)

- Simplify the use of results
- Teach people how to use the results

Showing the benefits

- What do we gain using a specific result?
  - performance
  - reliability
  - safety
  - predictability
  - efficiency
  - faster design

How much?

- Can you quantify in terms of money?

In mass productions every dollar
you save can be worth a million $.

Convincing people

First of all convince yourself.

- Identify critical scenarios (may be rare, but dangerous)

- Build working systems around critical scenarios

- Show that your method is able to deal with critical situations, so it's worth paying extra overhead.

A working demo is not enough

"Look! It's working using a RT kernel"

"A spoon in the bottle keeps Champagne bubbly"

What it is missing is **falsification**

Retis
Real-Time Systems Laboratory

What you have to show

"Look! It's NOT working under Windows"

"But it perfectly works under Erika Enterprise"

Identify at least a critical situation in which this happens.

Simplify

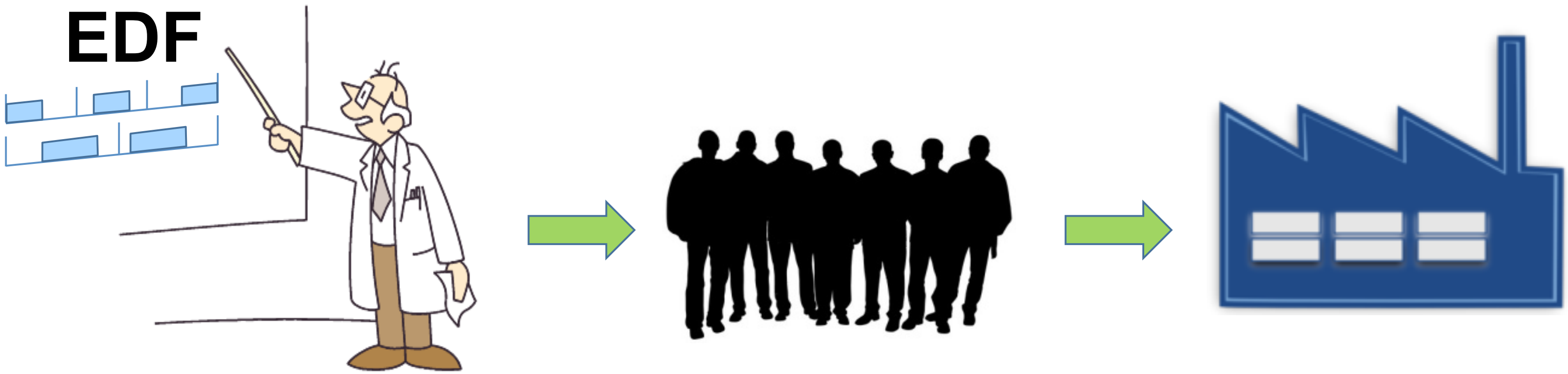
If you want to have a big impact in the society, then

seek for simplicity

- However, achieving simplicity is very difficult, because it implies
  - deep understanding
  - distilling ideas
  - beauty & elegance

- Hence, it takes **time**, **effort**, and **skill**

Teach



Retis
Real-Time Systems Laboratory

Read the literature

- To know the most recent results
- To avoid re-discovering the wheel

- Lots of results produced in the last 30 years

→ TCRTS Technical Committee on Real-Time Systems

repository of seminal papers

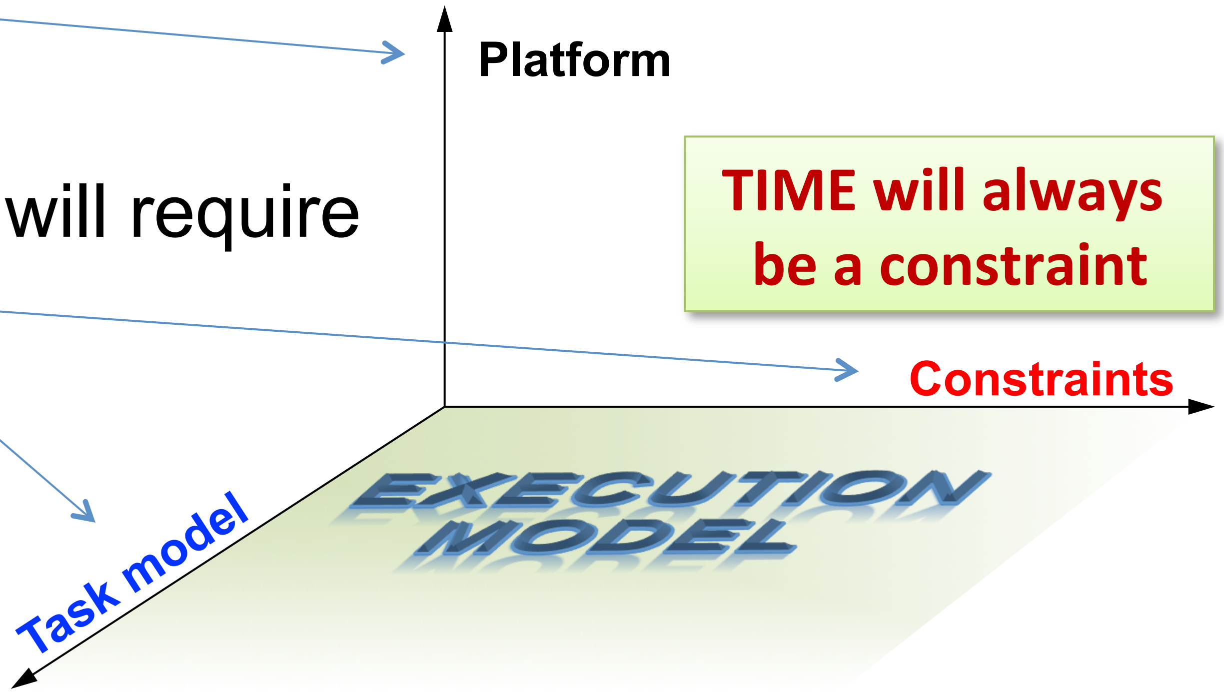
How about the future of RTS?

Will RTS become obsolete and reach a dead end?

How about the future of RTS?

New technologies will provide
new platforms

New applications will require
new models



TIME will always be a constraint

Just keep your models realistic and up-to-date

How about the future of RTS?

For the same reason

RTSS should keep its own
<u>identity</u> and <u>name</u>

Including new topics is good and necessary,
but following the wind and buzzwords is dangerous.

This year RTSS celebrates the **35th** anniversary.

Happy Birthday and Long Life