SAPIENZA
UNIVERSITÀ DI ROMA

Implementazione di sistemi real time

Automazione

Vincenzo Suraci

STRUTTURA DEL NUCLEO TEMATICO

- HARDWARE ABSTRACTION LAYER
- IMPLEMENTAZIONE EVENT-DRIVEN
- IMPLEMENTAZIONE TIME-DRIVEN
- SISTEMI DI AUTOMAZIONE REAL TIME
- SISTEMI OPERATIVI REAL TIME

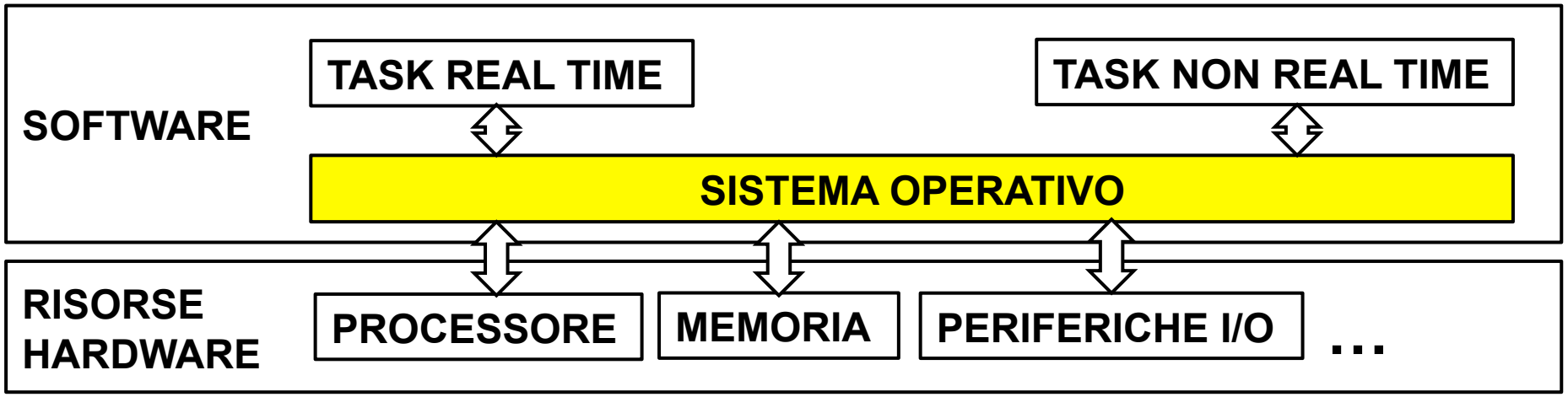
HARDWARE ABSTRACTION LAYER (HAL)

SISTEMA OPERATIVO

Un qualsiasi **sistema di controllo real time** è oggigiorno **implementato** via **software**.

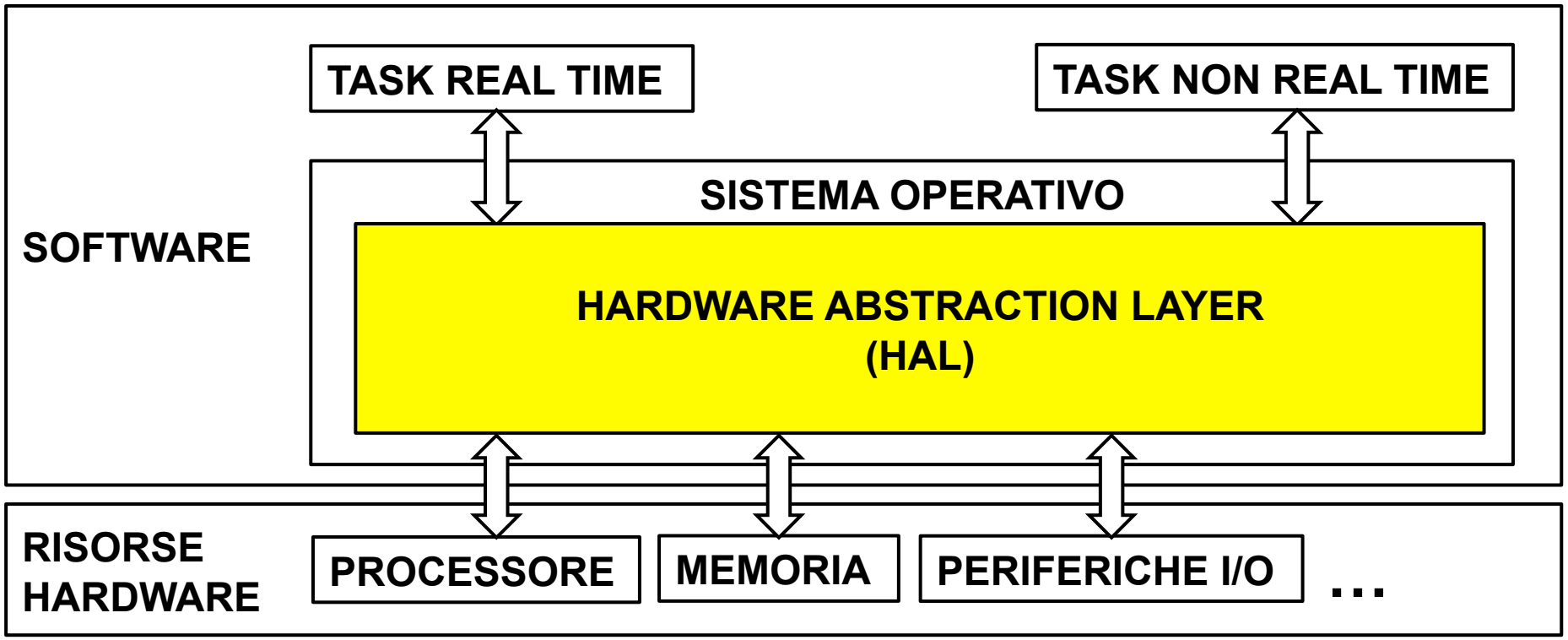
A fare da collante tra il livello **software** che implementa la logica del sistema di controllo e le **risorse hardware** controllate, vi è il **sistema operativo**.

Il **sistema operativo** è **l'interfaccia software** che permette di gestire i **task real time e non real time** che devono essere eseguiti dal **processore**.



HARDWARE ABSTRACTION LAYER

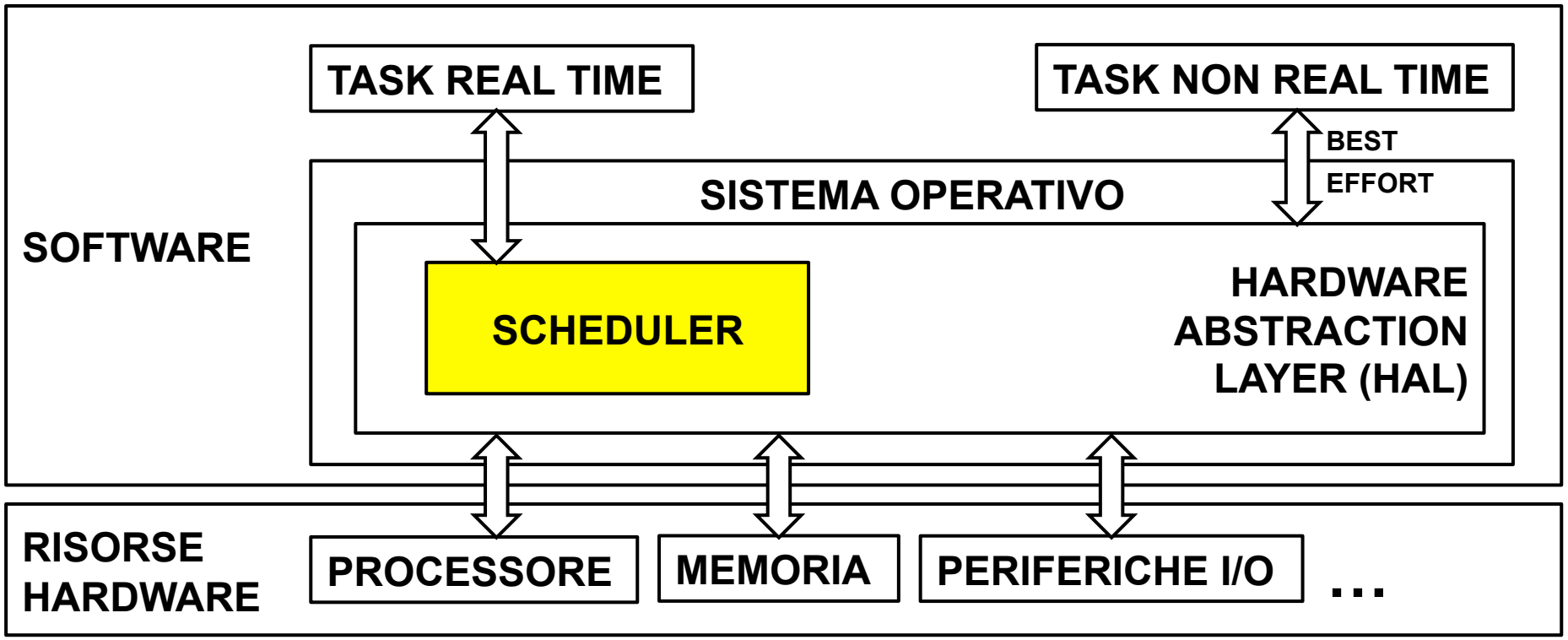
Per disaccoppiare il **sistema operativo** dalle infinite **possibili combinazioni di risorse hardware** viene introdotto un componente chiamato **HARDWARE ABSTRACTION LAYER**.



HARDWARE ABSTRACTION LAYER

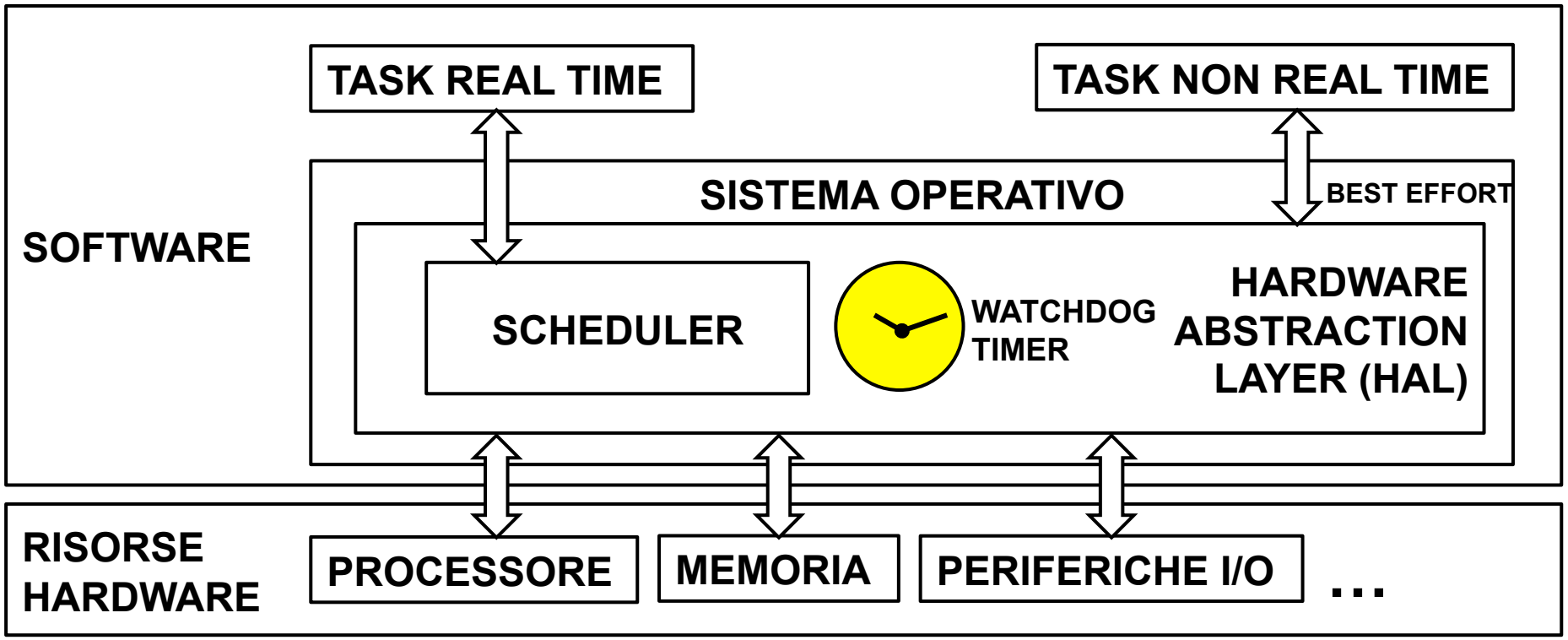
I task NON REAL TIME vengono gestiti dal HAL con politica **BEST EFFORT**.

I task REAL TIME vengono gestiti attraverso lo **SCHEDULER** che ospita uno degli algoritmi di scheduling studiati.



HARDWARE ABSTRACTION LAYER

Il **sistema operativo** controlla **periodicamente** che le **deadline dei task real time** vengano rispettate attraverso un **WATCHDOG TIMER**. Allo **scadere del timer** se una **deadline** è **scaduta**, l'anomalia è segnalata e viene eseguita una **routine di emergenza**.



IMPLEMENTAZIONE EVENT-DRIVEN

EVENT DRIVEN

DEFINIZIONE

Un **sistema operativo** si dice **EVENT DRIVEN** se esso è in grado di **schedulare un task** (scartandolo, mettendolo in coda o mandandolo subito in esecuzione) **nello stesso istante** in cui si verifica l'evento che lo ha **attivato**.

EVENT DRIVEN

VANTAGGI

- INTUITIVO: ogni task viene mandato allo scheduler non appena l'evento che lo attiva occorre
- SEMPLICE DA USARE: ad ogni task può essere **associata una priorità** e l'algoritmo di scheduling pensa a soddisfare anche vincoli hard real time

SVANTAGGI

- COMPLESSO DA REALIZZARE: **definire un algoritmo di scheduling** generale che risolva il problema della programmazione concorrente **è difficile ed oneroso** se non si conoscono a priori le **caratteristiche dei task in ingresso**

IMPLEMENTAZIONE TIME-DRIVEN

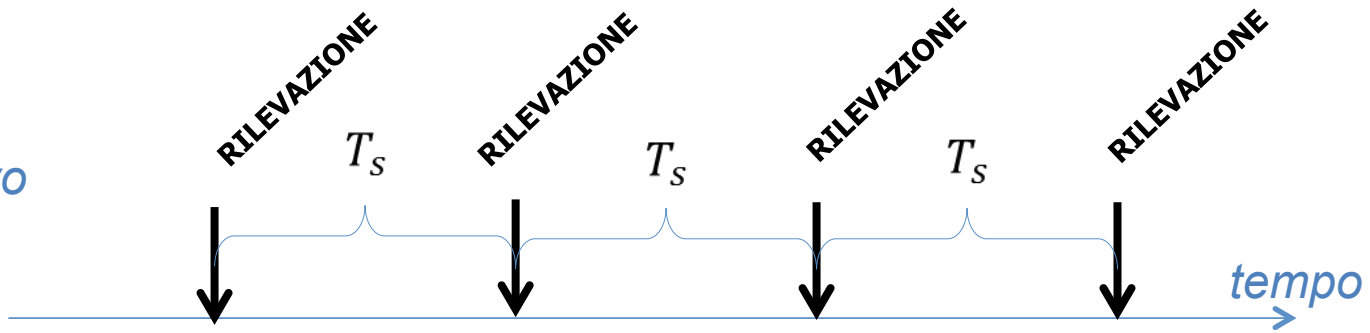
TIME DRIVEN

Un approccio **puramente event-driven** è in realtà **irrealizzabile** se l'unità di elaborazione è di tipo **digitale** e quindi intrinsecamente **quantizzata nel tempo**.

Un approccio realizzabile consiste nel **rilevare periodicamente** l'occorrenza di eventi e di gestire di conseguenza i relativi task. Tale approccio è detto **TIME DRIVEN**.

Il **periodo di tempo** che intercorre **tra due rilevazioni** consecutive è detto **PERIODO DI RILEVAZIONE** (o **PERIODO DI SCANSIONE**) T_s

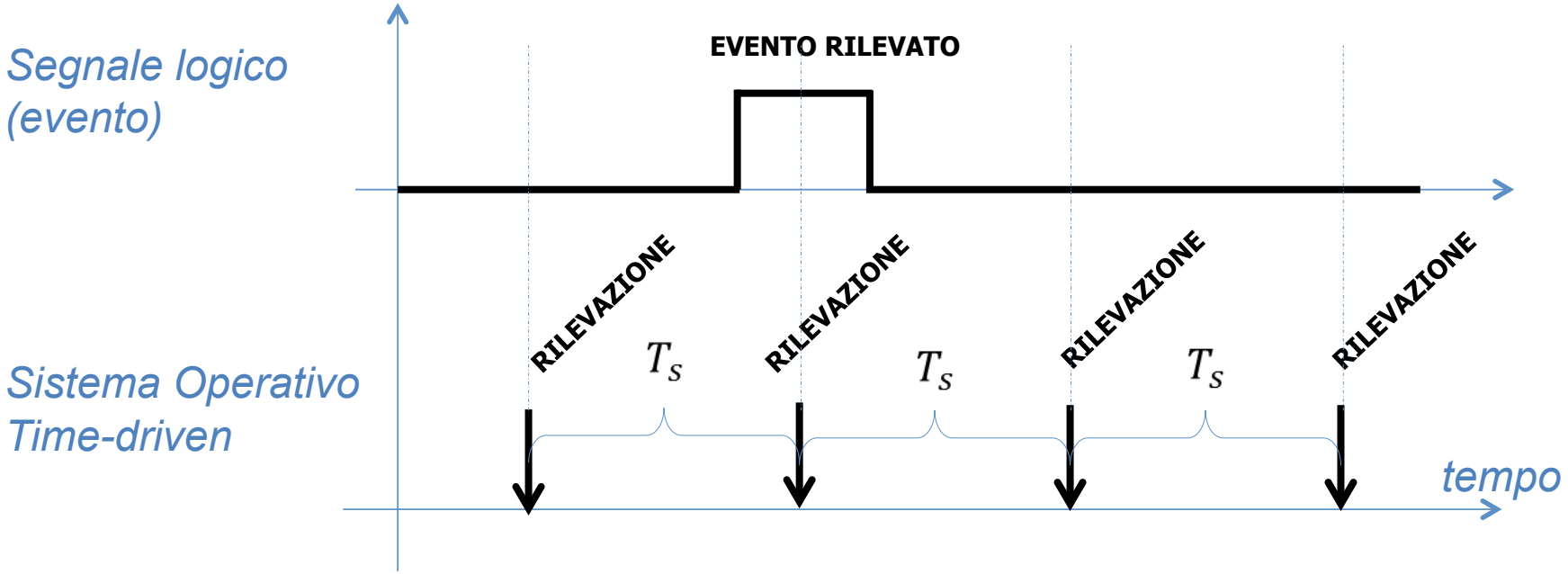
Sistema Operativo Time-driven



TIME DRIVEN

Un sistema di controllo TIME DRIVEN deve gestire necessariamente le problematiche relative alla **GESTIONE SINCRONA** di **EVENTI ASINCRONI** (che attivano i task).

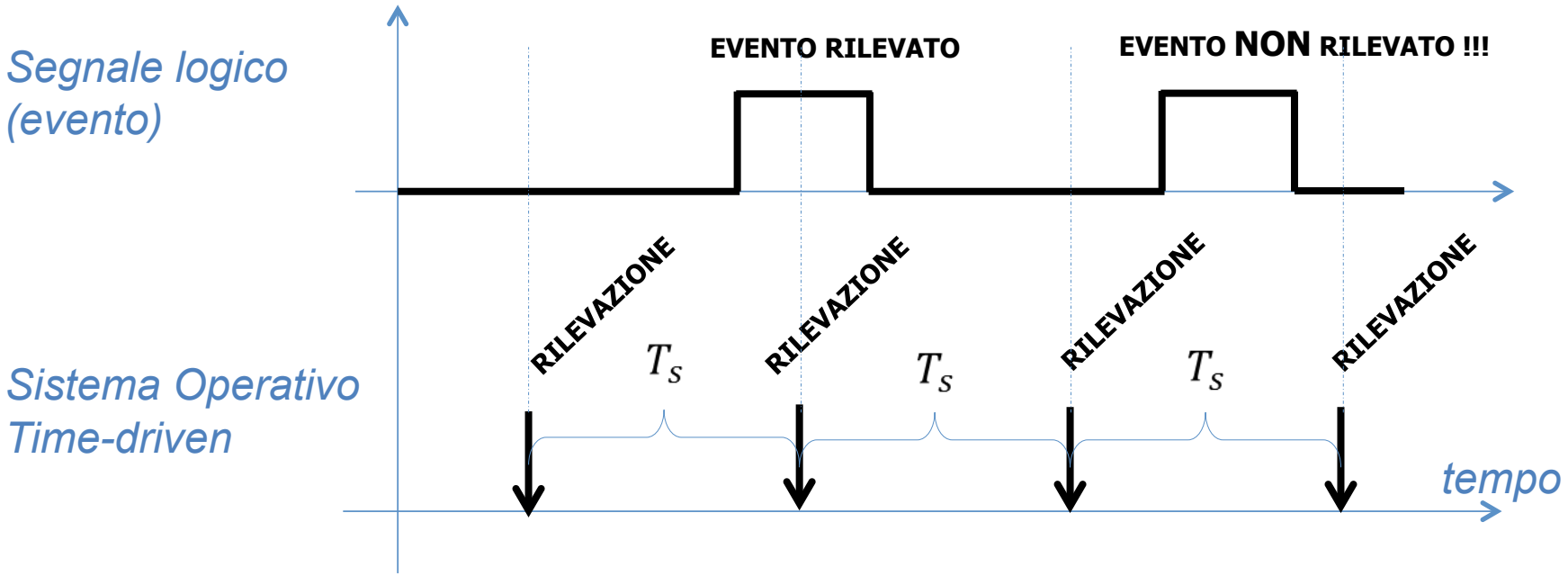
In un sistema di controllo **DIGITALE**, un **EVENTO** può essere associato al **valore logico** di un **variabile binaria** (SEGNALE LOGICO).



TIME DRIVEN

PROBLEMA 1 – OSSERVABILITÀ DEGLI EVENTI

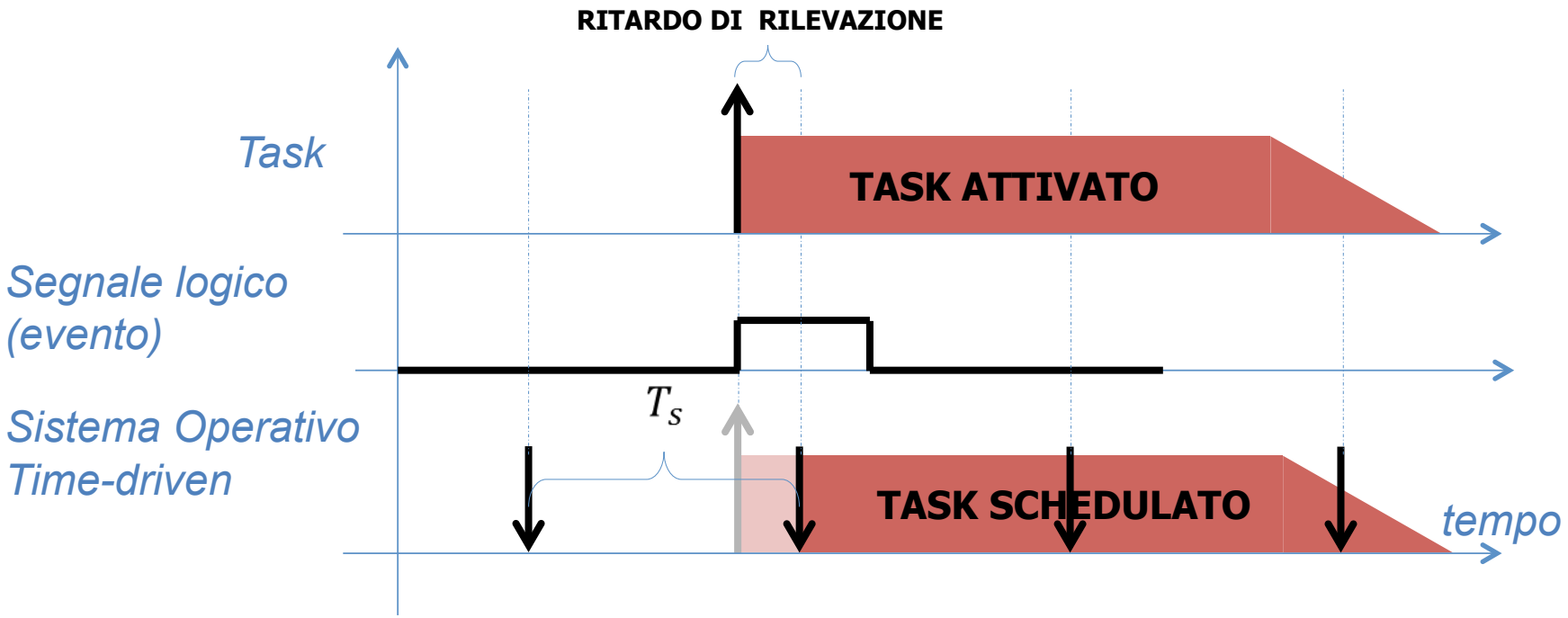
In un sistema di controllo **time driven** un **evento** può **NON ESSERE OSSERVABILE**. In particolare ciò può avvenire solo se il segnale logico associato all'evento rimane attivo per un tempo **inferiore del periodo di rilevazione**.



TIME DRIVEN

PROBLEMA 2 – RITARDO DI RILEVAZIONE

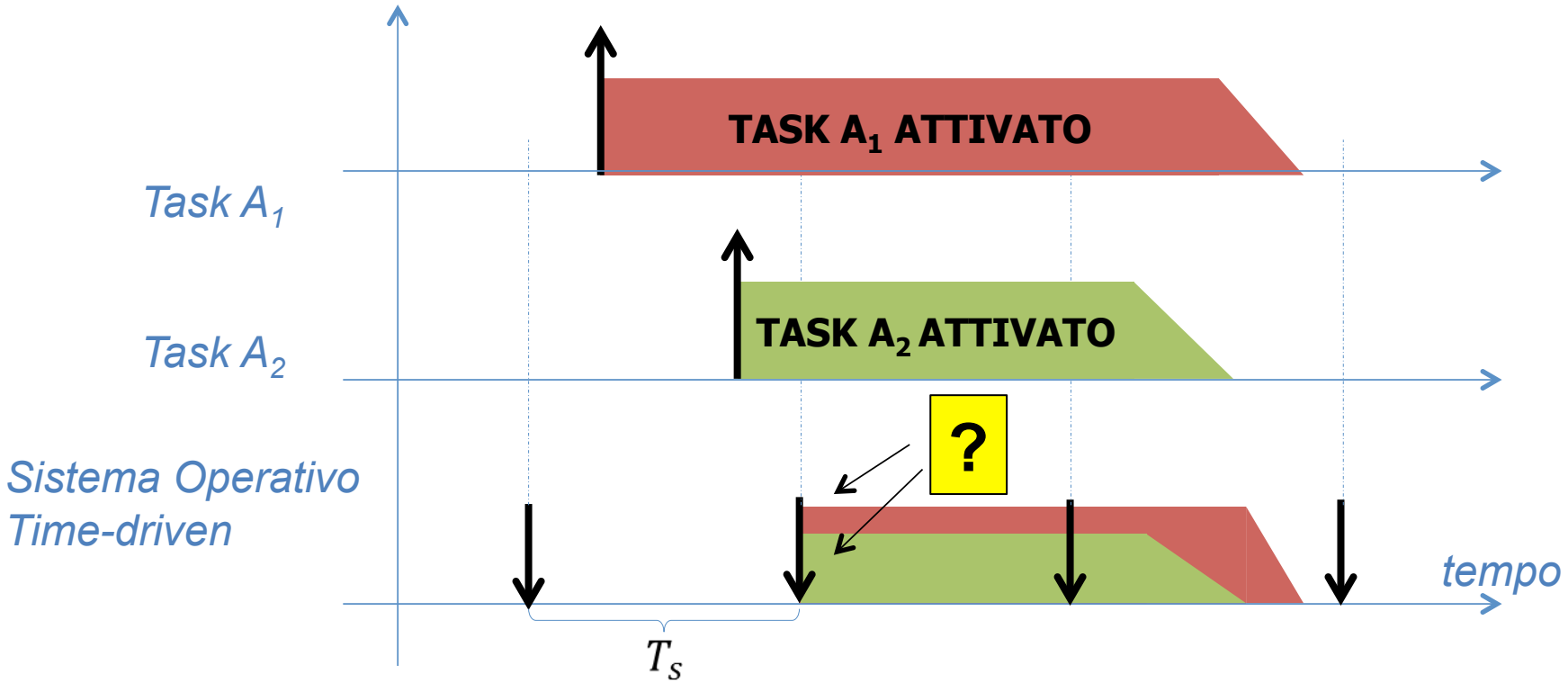
In un sistema di controllo **time driven** ogni occorrenza di un task è soggetta ad un **ritardo di rilevazione** che impatta necessariamente sullo **start time** del task.



TIME DRIVEN

PROBLEMA 3 – ORDINE DI OCCORRENZA

In un sistema di controllo **time driven**, l'**ordine** in cui si presentano due o più eventi occorrenti tra due rilevazioni successive **viene perso**.

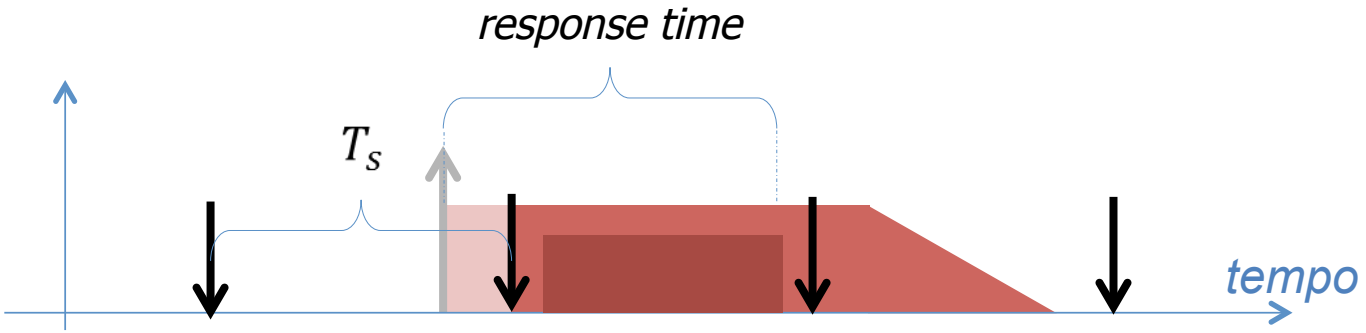


TIME DRIVEN

VANTAGGI

- SEMPLICE DA REALIZZARE: **è sufficiente abilitare un timer** per rilevare **periodicamente** l'occorrenza di **eventi**

- REATTIVITÀ: ipotizzando che l'elaborazione di qualsiasi task si concluda entro l'intervallo di tempo tra due istanti di rilevazione successivi, è possibile determinare il **limite superiore del response time** del sistema di controllo, pari a **due volte il periodo di rilevazione**

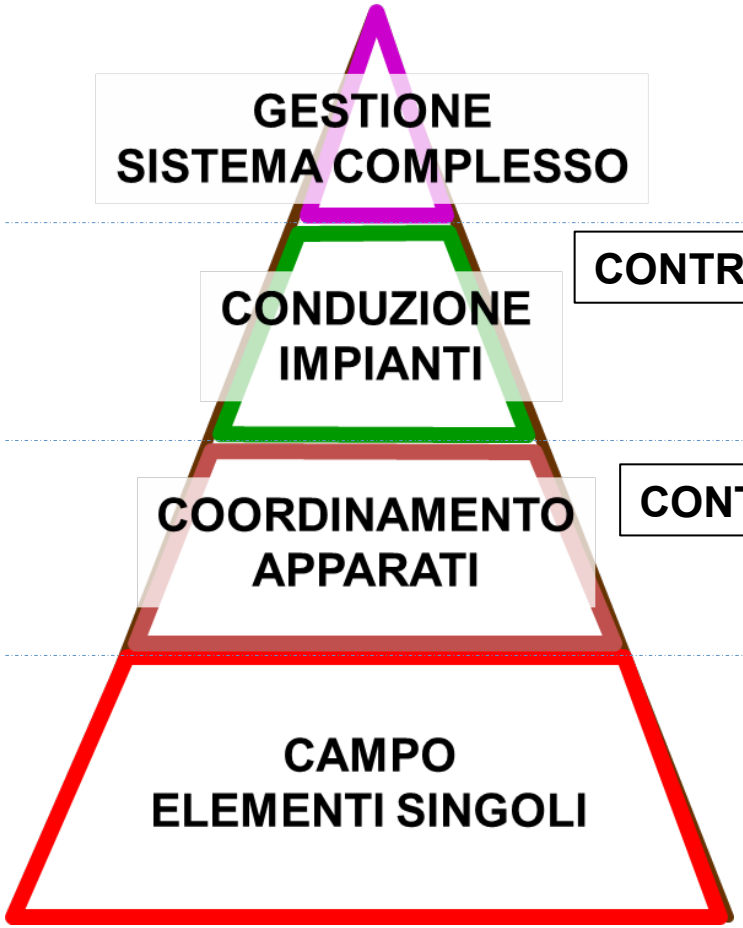


SVANTAGGI

- FLESSIBILITÀ: le problematiche evidenziate (osservabilità degli eventi, ritardo di rilevazione, ordine di occorrenza) rendono questo approccio poco flessibile

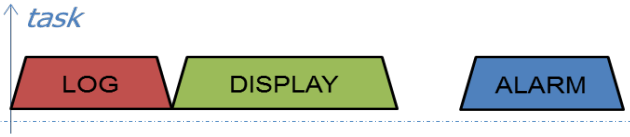
SISTEMI DI AUTOMAZIONE
REAL TIME

OBIETTIVO DEI SISTEMI DI AUTOMAZIONE REAL TIME



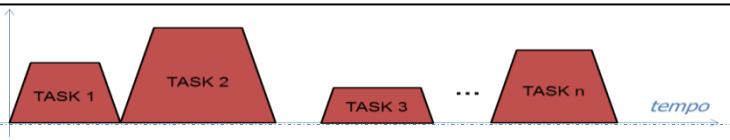
OBIETTIVO dei SISTEMI DI AUTOMAZIONE REAL TIME
in funzione della PIRAMIDE DELLA AUTOMAZIONE

CONTROLLO DI SEQUENZE LOGICHE PER LA SUPERVISIONE



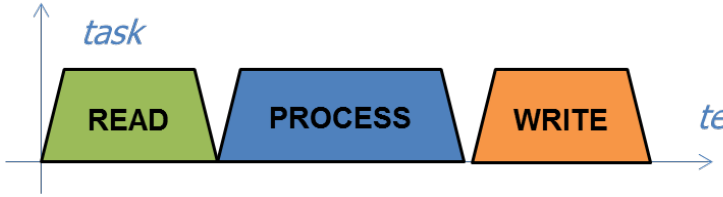
PERIODICI & APERIODICI

CONTROLLO DI SEQUENZE LOGICHE DI COORDINAMENTO



PERIODICI & APERIODICI

CONTROLLO DIGITALE DI VARIABILI ANALOGICHE



PERIODICI

REALIZZAZIONE DEI SISTEMI DI AUTOMAZIONE REAL TIME

REALIZZAZIONE dei SISTEMI DI AUTOMAZIONE REAL TIME in funzione della PIRAMIDE DELLA AUTOMAZIONE

EVENT DRIVEN

TIME DRIVEN

REALIZZAZIONE DEI SISTEMI DI AUTOMAZIONE REAL TIME

OSSERVAZIONE

A livello di **coordinamento** e **conduzione** sono presenti **task misti**, pertanto è intuitivo pensare di usare sistemi di controllo real time **event driven**.

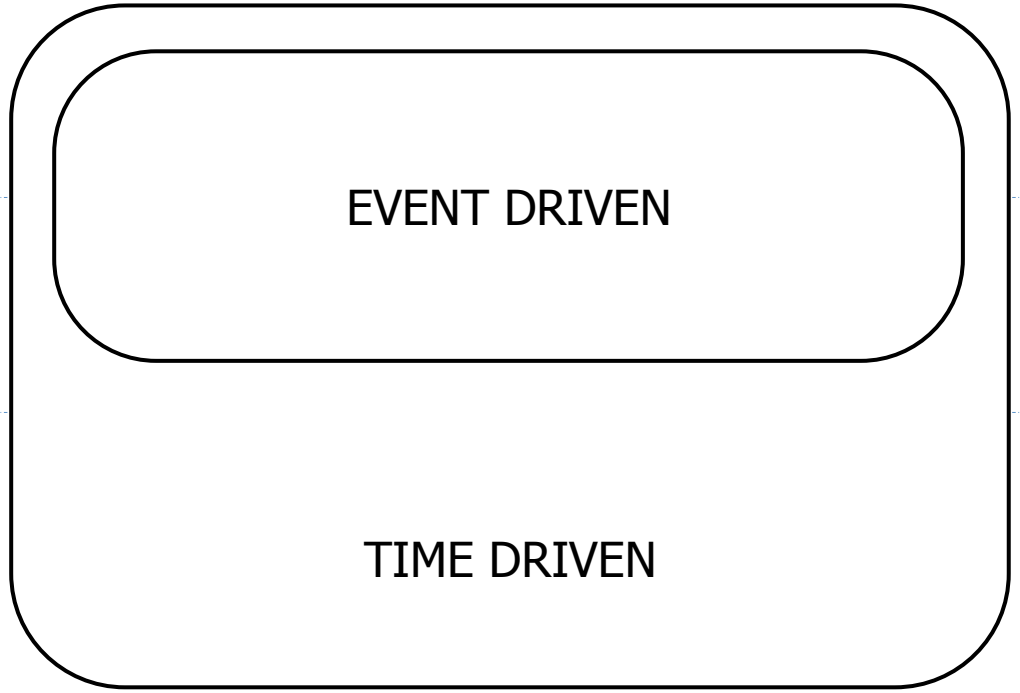
Ma abbiamo visto quanto sia:

- difficile forzare il sistema operativo ad essere costantemente pronto a rilevare un nuovo evento;
- complesso realizzare uno scheduling hard real time.

Pertanto **dal punto di vista implementativo** è molto **più conveniente** realizzare sistemi di controllo finalizzati all'Automazione **completamente time driven**.

REALIZZAZIONE DEI SISTEMI DI AUTOMAZIONE REAL TIME

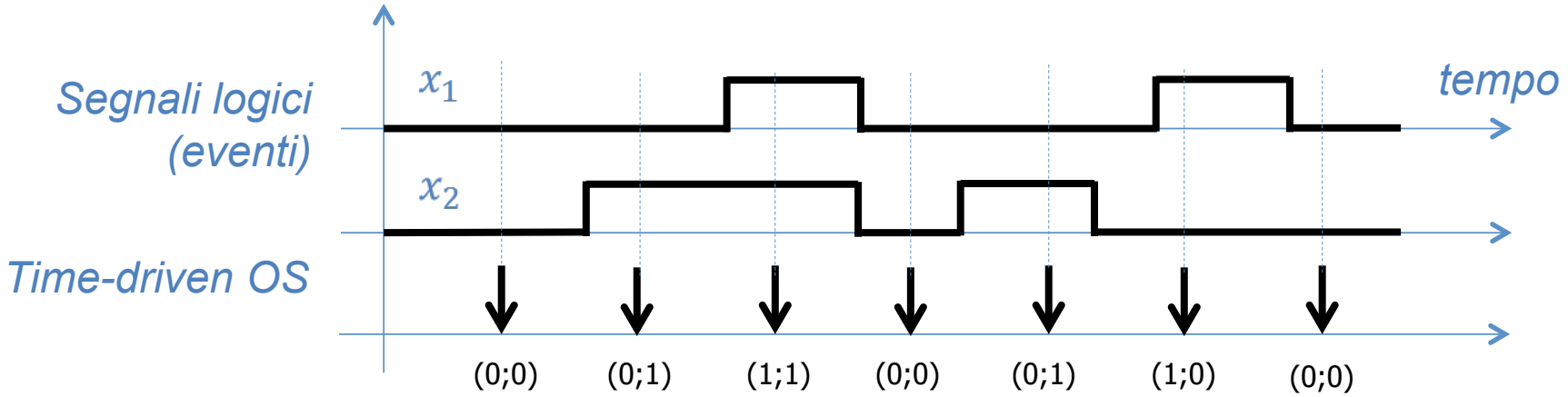
REALIZZAZIONE dei SISTEMI DI AUTOMAZIONE REAL TIME in funzione della PIRAMIDE DELLA AUTOMAZIONE



DA EVENT-DRIVEN A TIME-DRIVEN

OSSERVAZIONE

Un evento è un'entità asincrona (slegata quindi dal clock del sistema di controllo digitale) che **modifica lo stato** del sistema.



L'idea è quella di **fotografare periodicamente** (in maniera sincrona con il clock del sistema di controllo digitale) tale **stato** e **in base ad esso eseguire i task necessari**.

In tale modo si ottiene una **gestione SINCRONA e PERIODICA** di TUTTE le attività e quindi si aprono le porte ad una implementazione **completamente time-driven**.

DA EVENT-DRIVEN A TIME-DRIVEN

Abbiamo pertanto ricondotto il problema del controllo di un sistema di Automazione al problema di gestire n task periodici di periodo T_i $(i = 1,2, \dots, n)$ che includono task di livello di campo, di livello di coordinamento e di livello di conduzione.

IPOTESI 1

Il problema del **ritardo di rilevazione** può essere **mitigato diminuendo** opportunamente il **periodo di rilevazione** T_s. In particolare si pone:

$$T_s \leq \min_i(T_i)$$

CONSIDERAZIONE

Si potrebbe prendere T_s piccolo a piacere, ma questo aumenterebbe inutilmente la frequenza con cui il sistema di controllo verifica il cambio di stato.

DA EVENT-DRIVEN A TIME-DRIVEN

IPOTESI 2

Il problema del **ritardo di rilevazione** può essere **mitigato** ipotizzando che **ogni task** abbia una **deadline relativa pari almeno al doppio del periodo di rilevazione**:

$$D_i \geq 2T_s \quad \forall\, i = 1,2,\dots,n$$

IPOTESI 3

Ipotizziamo che la **somma dei tempi di calcolo degli n task sia inferiore al periodo di rilevazione**:

$$\sum_{i=1}^{n} C_i < T_s$$

DA EVENT-DRIVEN A TIME-DRIVEN

PROPOSIZIONE (senza dimostrazione)

Dato un sistema di Automazione composto da un sistema di controllo TIME DRIVEN con periodo di rilevazione T_s, e da n task periodici di periodo T_i $(i = 1,2,\dots,n)$ e computation time C_i $(i = 1,2,\dots,n)$ che rispettino le tre ipotesi:

$$\begin{cases} T_s \leq \min_i(T_i) \\ D_i \geq 2T_s \quad \forall\, i = 1,2,\dots,n \\ \sum_{i=1}^{n} C_i < T_s \end{cases}$$

La schedulazione dei task può **SEMPRE** avvenire usando un algoritmo **TIMELINE SCHEDULING** scegliendo come **MINOR CYCLE il PERIODO DI RILEVAZIONE**.

DA EVENT-DRIVEN A TIME-DRIVEN

OSSERVAZIONE

Al sistema di controllo TIME-DRIVEN possono essere accostate strategie di scheduling BEST EFFORT di task completamente aperiodici (ad es. richieste manuali di informazioni sullo stato della macchina tramite Human Machine Interface).

Si può accostare al sistema di controllo TIME-DRIVEN una strategia di SERVIZIO IN BACKGROUND.

ESEMPIO

PROBLEMA

Dato un problema di Automazione con sistema di controllo digitale TIME DRIVEN composto da 3 task periodici:

- Lettura ingressi (T_1= 8 t.u. , C_1= 1 t.u.)
- Elaborazione azioni di intervento (T_2= 12 t.u. , C_2= 1 t.u.)
- Attuazione (T_3= 16 t.u. , C_3= 1 t.u.)

e da un task aperiodico:

- Richiesta aggiornamento HMI ($a_4(1)$ = 3 t.u., $C_4(1)$ = 15 t.u., $D_4(1)$ = 40 t.u.)

mostrare uno schema di timeline scheduling + servizio in background che risolva il problema dato.

ESEMPIO cont'd

SVOLGIMENTO

Verifichiamo la condizione NECESSARIA per la schedulabilità dei task periodici:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{1}{8} + \frac{1}{16} + \frac{1}{12} = \frac{6+3+4}{48} = \frac{13}{48} = 0{,}27083\overline{3} < 1$$

Verifichiamo la condizione SUFFICIENTE per i sistemi TIME DRIVEN:

$$T_s = MINOR\ CYCLE = MCD(8,16,12) = 4\ t.u.$$

$$T_s = 4 \leq \min_i(T_i) = 8\ t.u.$$

$$D_i = T_i \geq 2T_s = 8 \quad \forall\, i = 1,2,3$$

$$\sum_{i=1}^{3} C_i = 3 < 4 = T_s$$

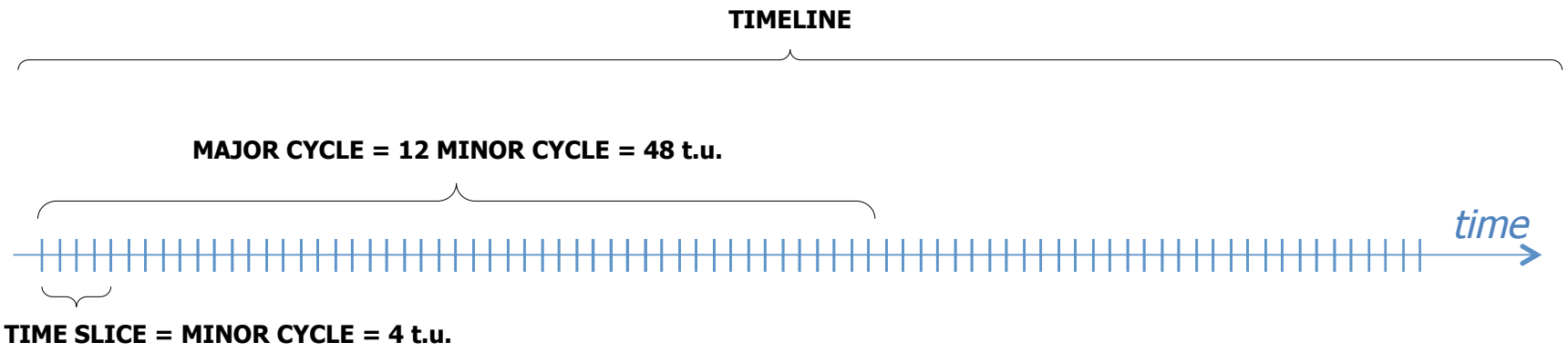
ESEMPIO cont'd

Passiamo quindi a tracciare la TIMELINE e i differenti TIMESLICE che la compongono.

Calcoliamo pertanto il MINOR CYCLE (pari alla durata del TIMESLICE) e il MAJOR CYCLE (che definisce la periodicità dell'algoritmo di TIMELINE SCHEDULING).

$$MINOR\ CYCLE = MCD(8,16,12) = 4\ t.u.$$

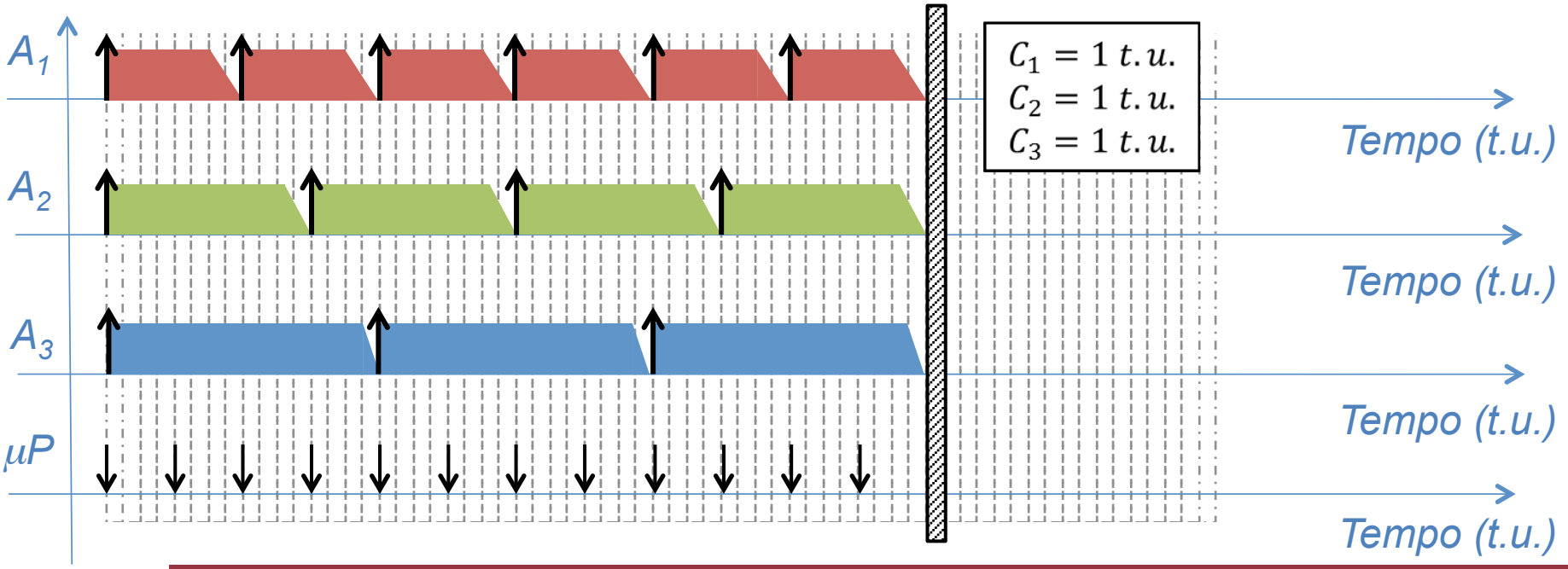
$$MAJOR\ CYCLE = mcm(8,16,12) = 48\ t.u.$$



ESEMPIO cont'd

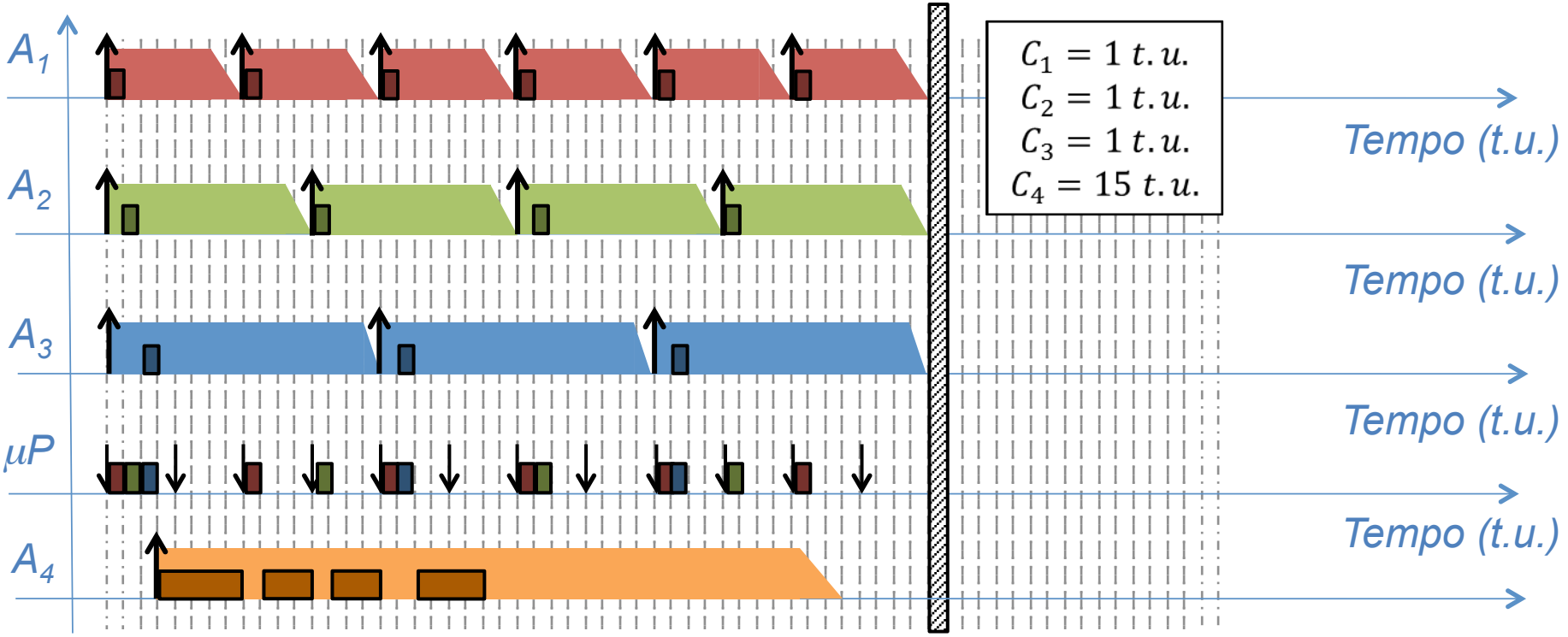
Tracciamo il diagramma temporale dei 3 task periodici ed identifichiamo la soluzione, notando che:

1. Il task A_1 dovrà ripetersi MAJOR CYCLE / T_1 = 6 volte in un MAJOR CYCLE
2. Il task A_2 dovrà ripetersi MAJOR CYCLE / T_2 = 4 volte in un MAJOR CYCLE
3. Il task A_3 dovrà ripetersi MAJOR CYCLE / T_3 = 3 volte in un MAJOR CYCLE



ESEMPIO cont'd

Troviamo una soluzione al problema:



SISTEMI OPERATIVI REAL TIME

Consumer Electronics OS

I sistemi operativi più diffusi (Windows, Linux, Mac OS) **non sono adatti** per gestire sistemi di controllo **real time**.

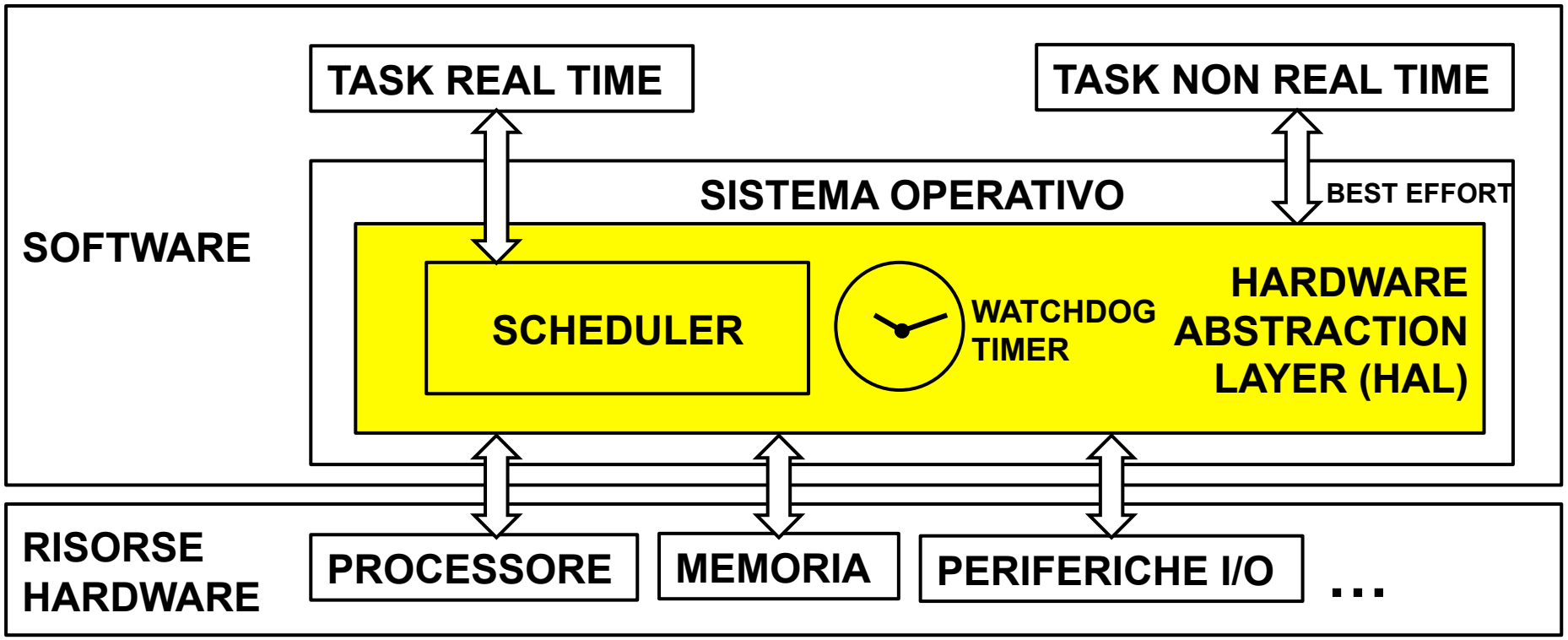
Il problema principale risiede nella **impossibilità di determinare il massimo tempo di esecuzione di un task** (processo o thread che sia).

Un computer **non industriale** è equipaggiato con **risorse** che **bloccano la CPU** e rendono difficilissimo gestire il **determinismo** dello scheduler implementato nel Kernel:

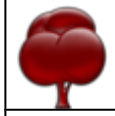
- Periferiche di I/O (Universal Serial Bus – USB)
- DMA (Direct Memory Access) dell'Hard Disk
- CACHE della CPU (quando si svuota la CPU non può lavorare)
- Memoria Virtuale (paging)
- IRQ (Interrupt Request) da parte di periferiche PCI
- ACPI (Advanced Configuration and Power Interface) cambia la frequenza della CPU

Sistemi Operativi Real Time

Per rendere un Sistema Operativo Real Time **è necessario mettere mano al codice** del suo **scheduler** e del suo **HAL** (Hardware Abstraction Layer). Ma questo è possibile solo se il Kernel del sistema operativo è **Open Source**.



Esempi di Sistemi Operativi Real Time – NON COMMERCIALI

	Nome	Caratteristiche
	ChibiOS/RT	Open Source ; Sistemi Embedded
	TinyOS	Open Source; Wireless Sensor Nodes
	RTAI Linux	Open Source (italiano); Computer Industriali; Linux based
	BeRTOS NON SOLO KERNEL	Open Source (italiano); Sistemi Embedded; Arduino
	freeRTOS	Open Source; Cross platform
	Ethernut	Open Source; Sistemi embedded dedicati
	milOS	Open Source; Sistemi embedded

BIBLIOGRAFIA

Sezione 2.5 e 2.6

TITOLO

Sistemi di automazione industriale

Architetture e controllo

AUTORI

Claudio Bonivento

Luca Gentili

Andrea Paoli

EDITORE

McGraw-Hill