Corso di Robotica 1

Architetture di governo
Programmazione

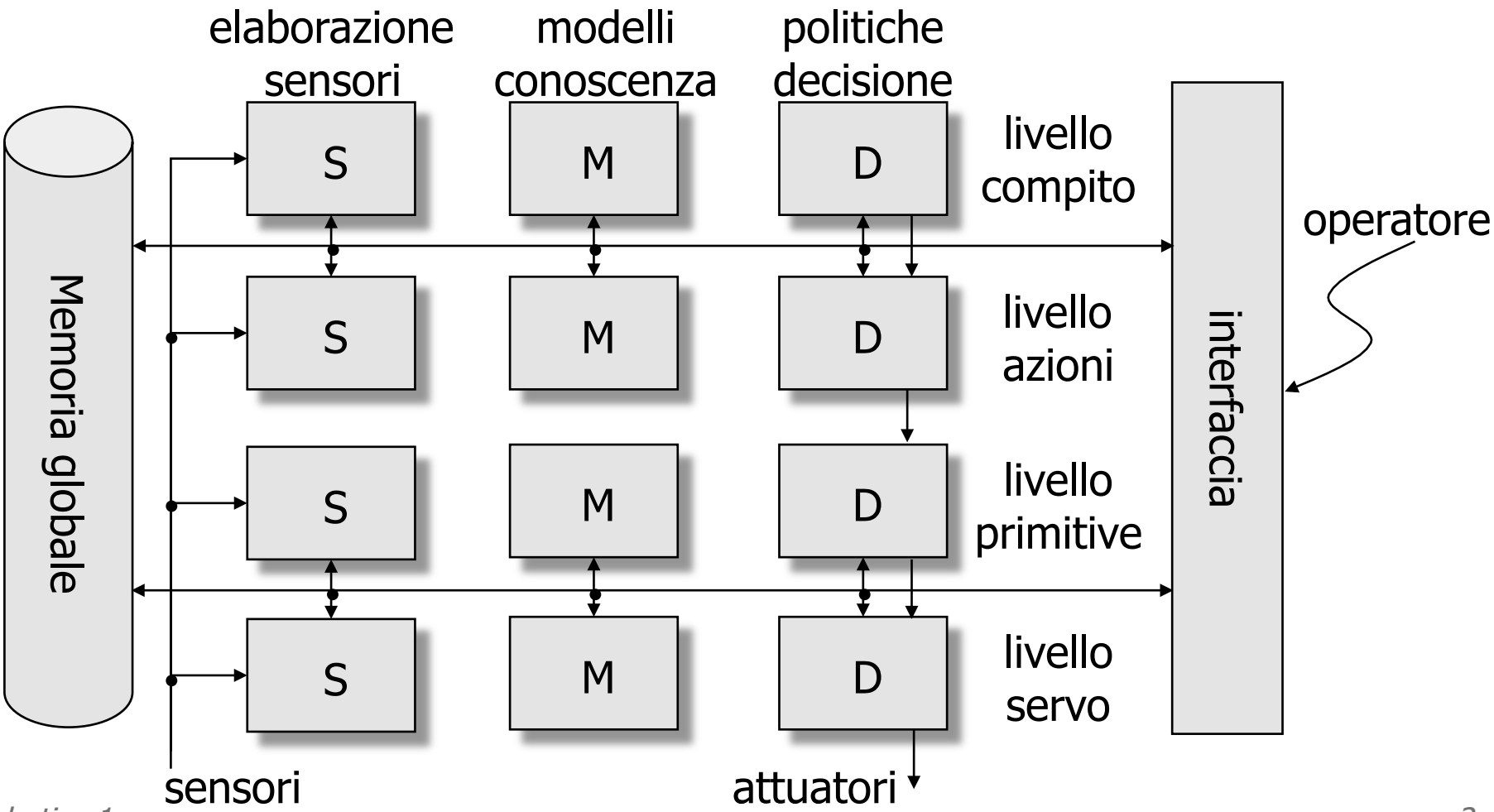
Prof. Alessandro De Luca

DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

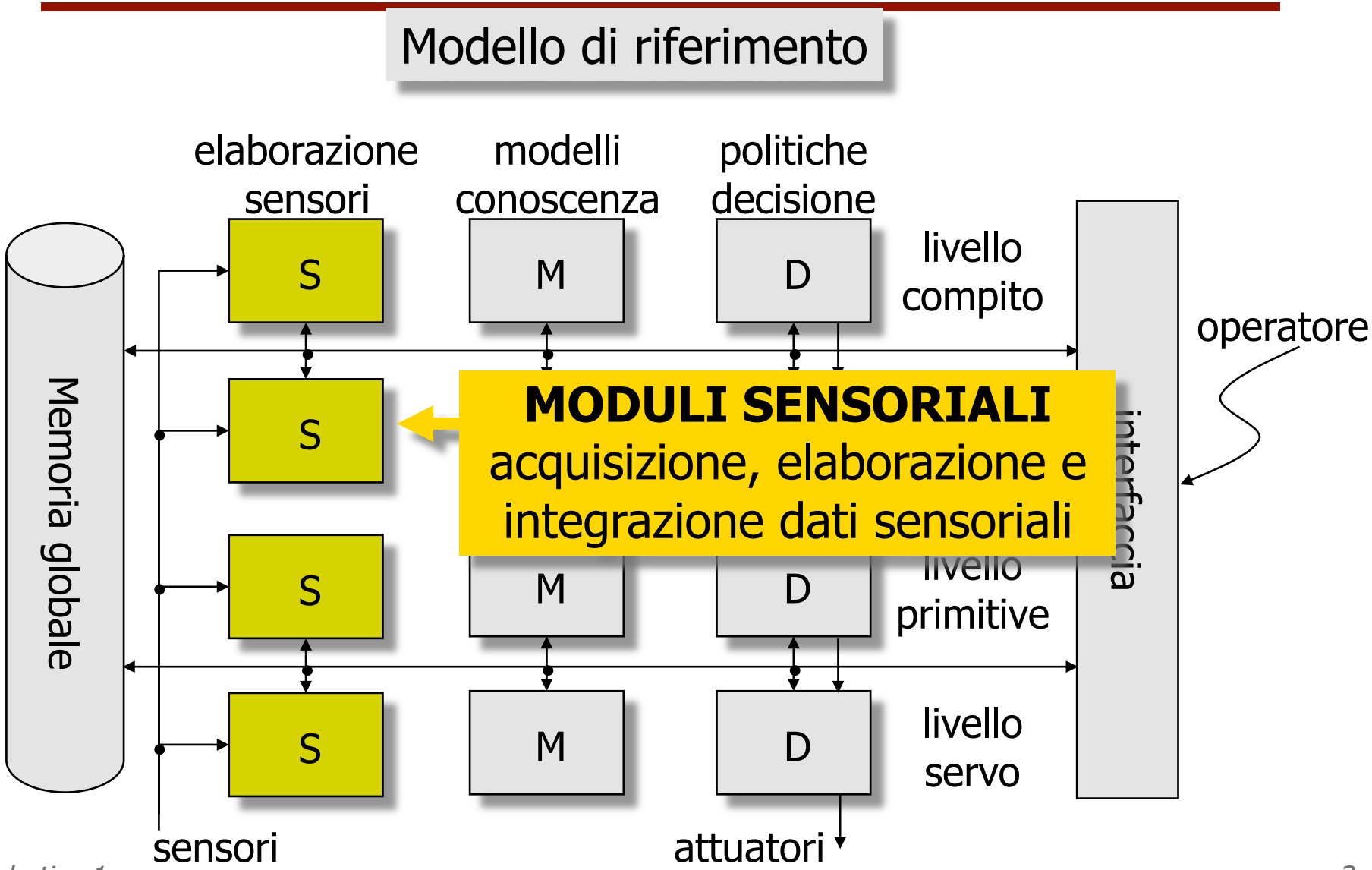
SAPIENZA
UNIVERSITÀ DI ROMA

Architettura funzionale di governo

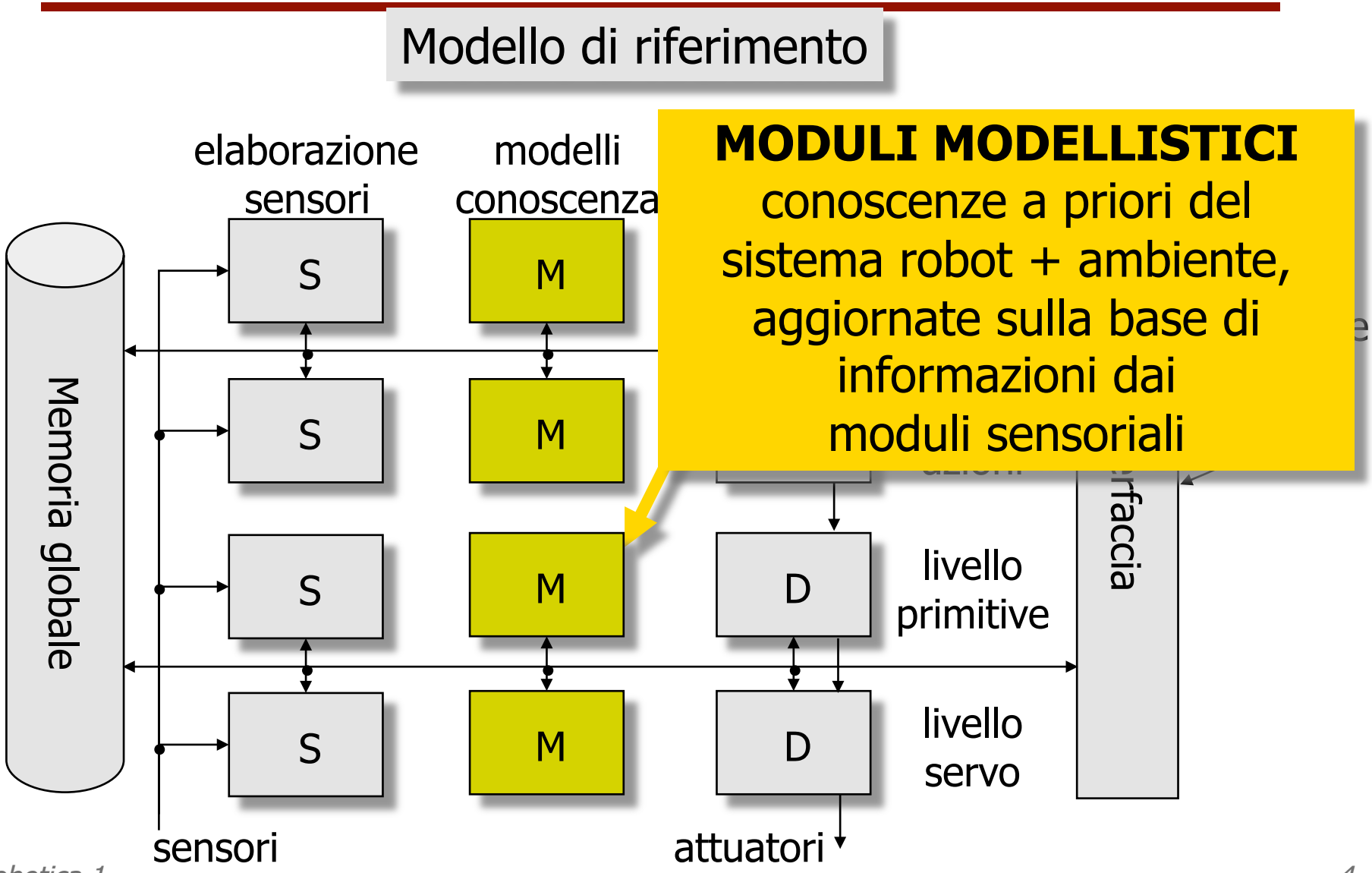
Modello di riferimento



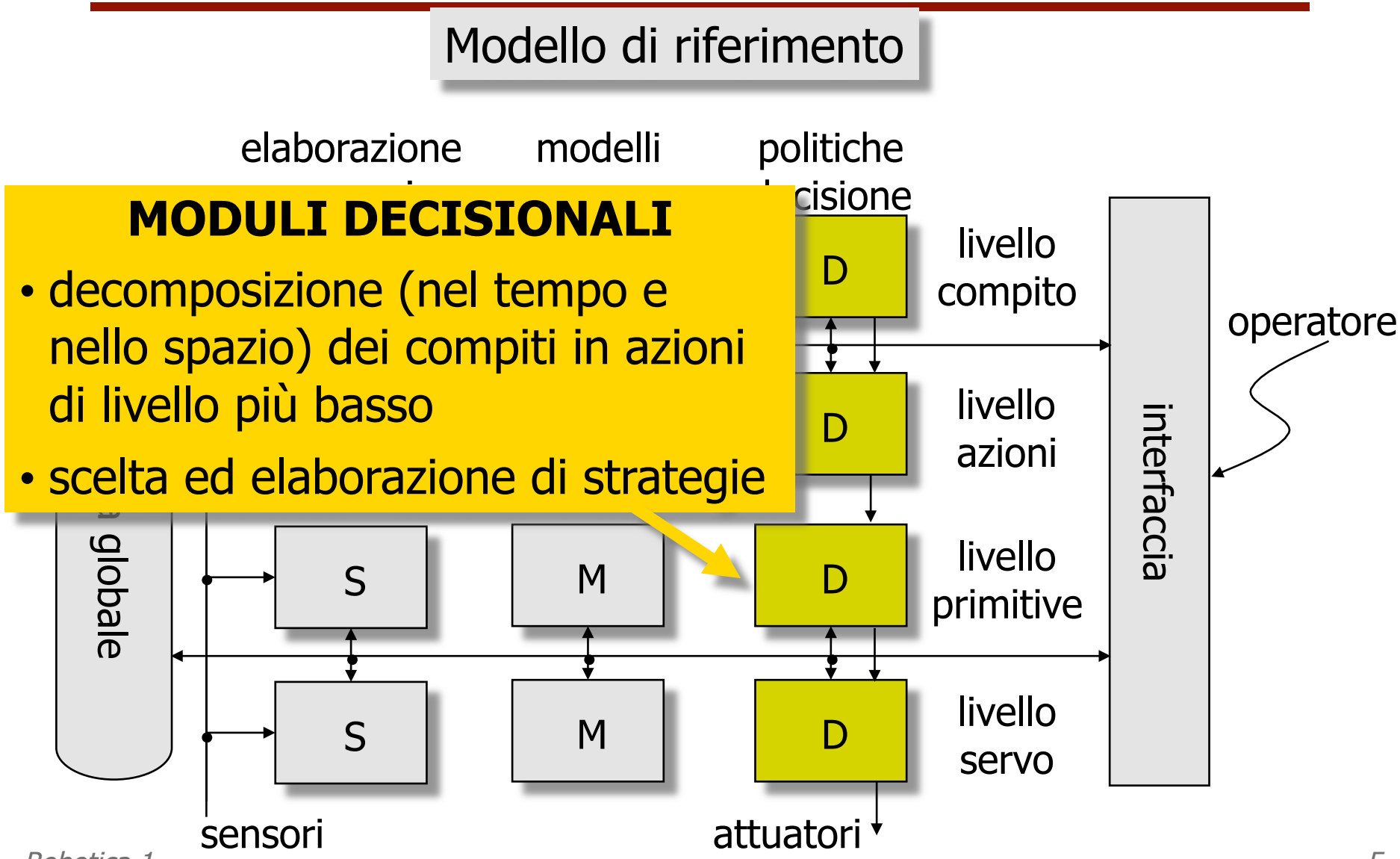
Modello di riferimento: Moduli



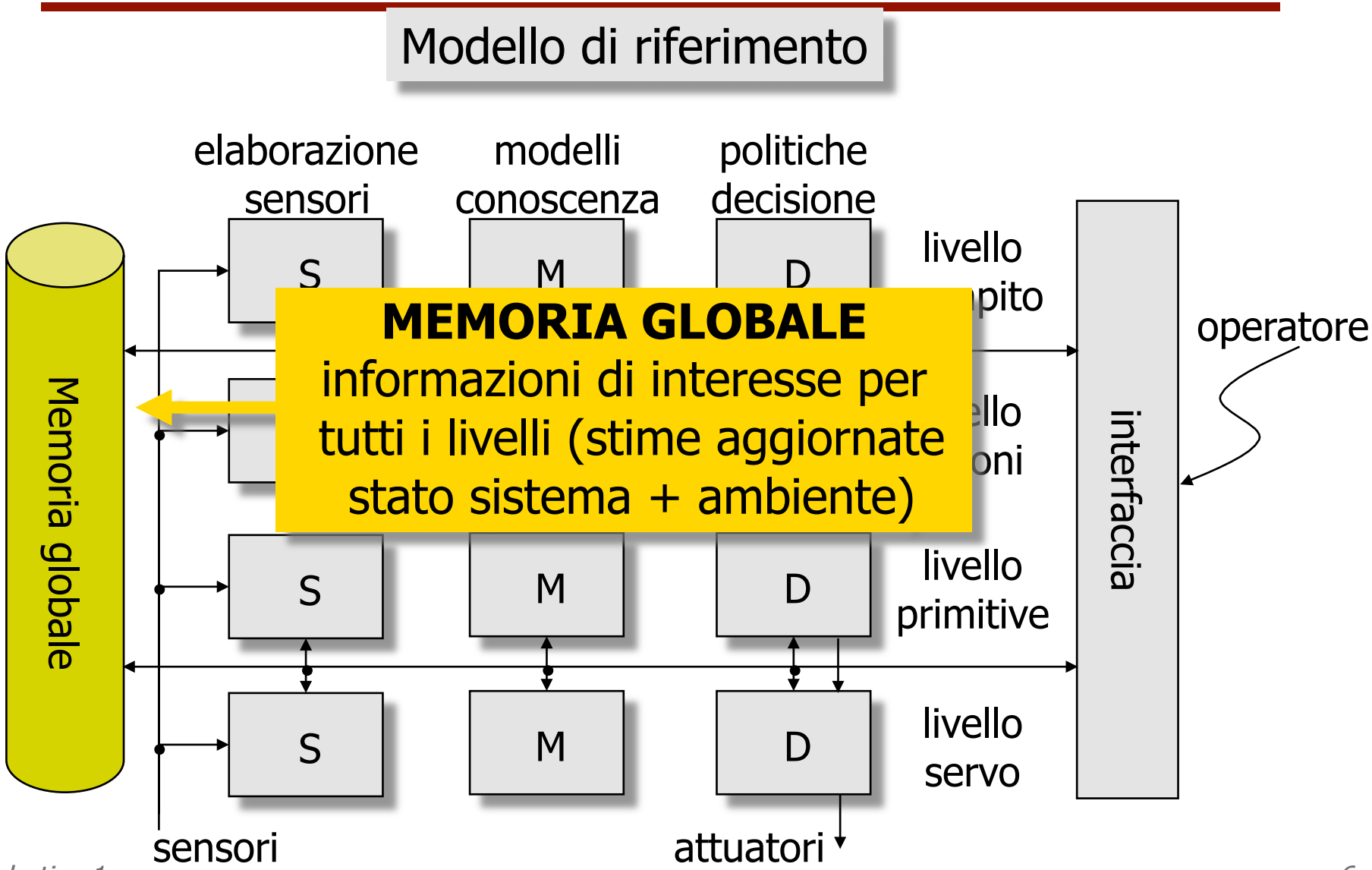
Modello di riferimento: Moduli



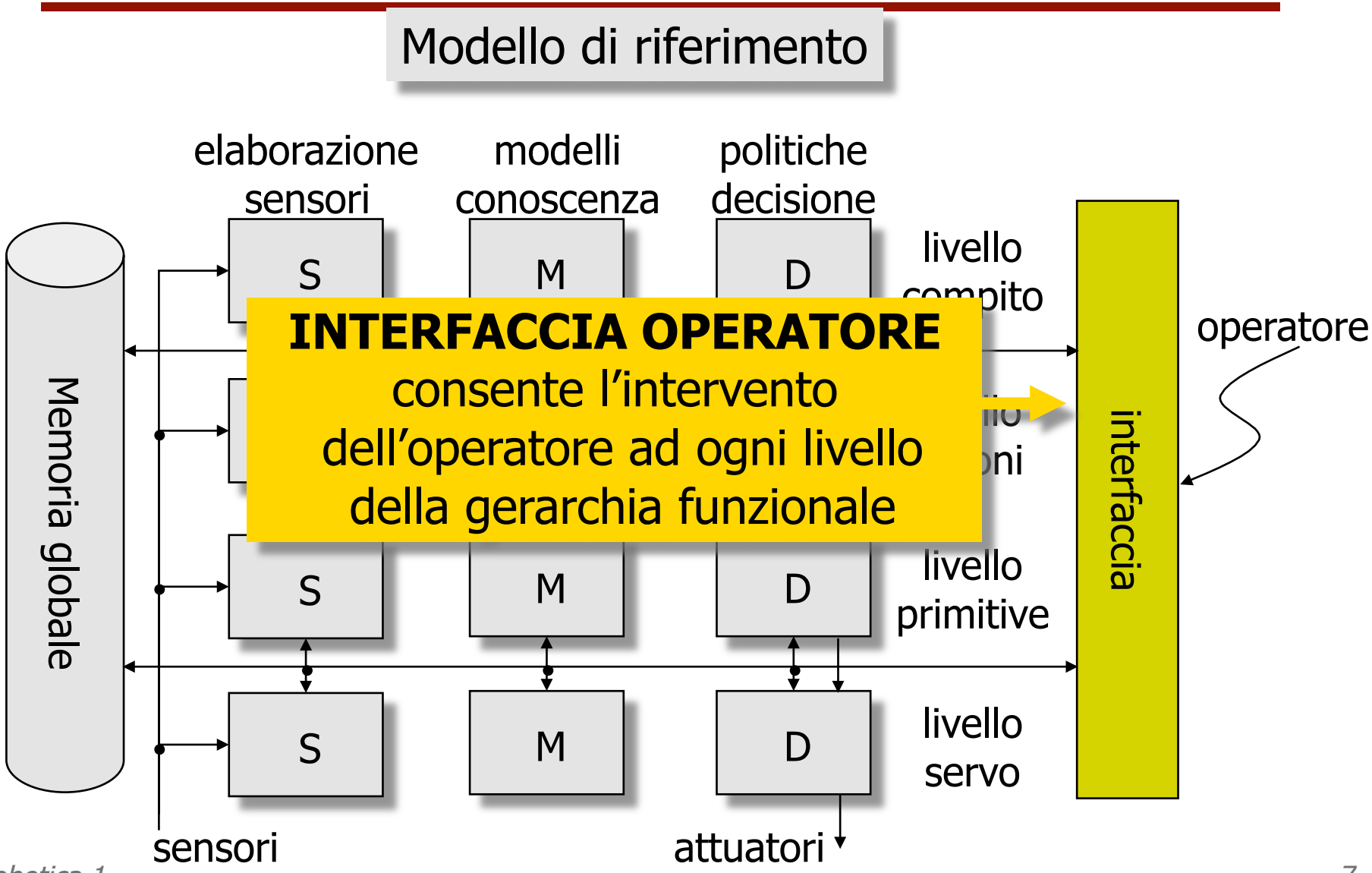
Modello di riferimento: Moduli



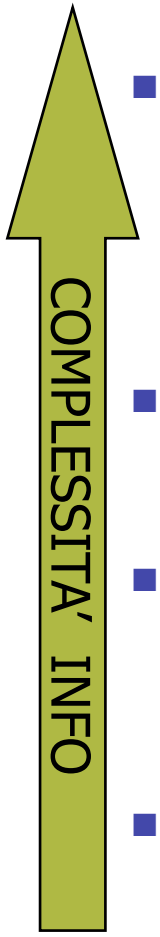
Modello di riferimento: Moduli



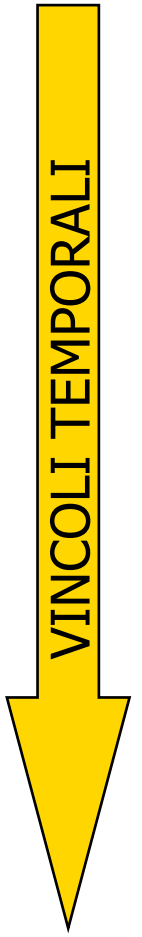
Modello di riferimento: Moduli



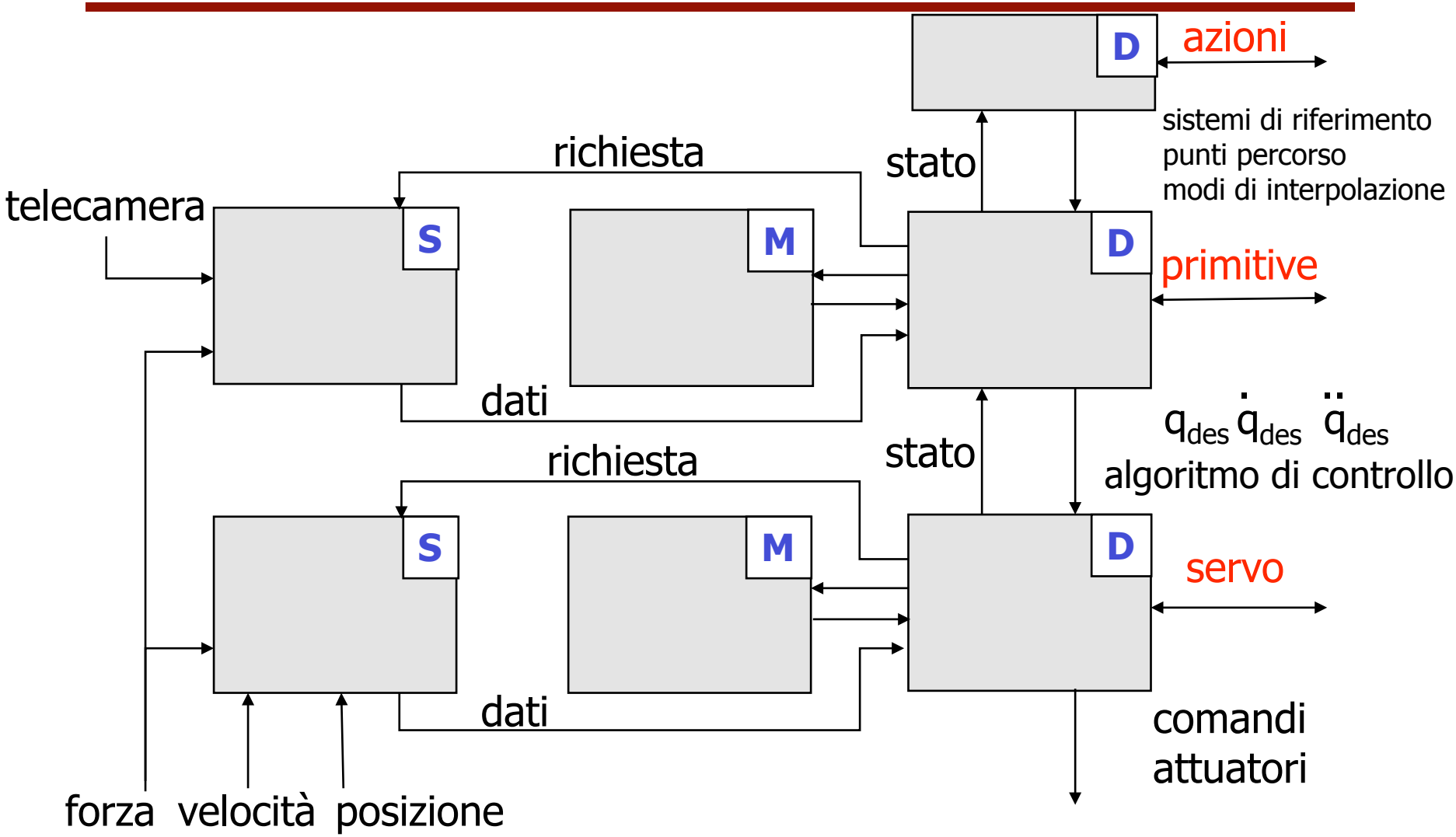
Modello di riferimento: Livelli



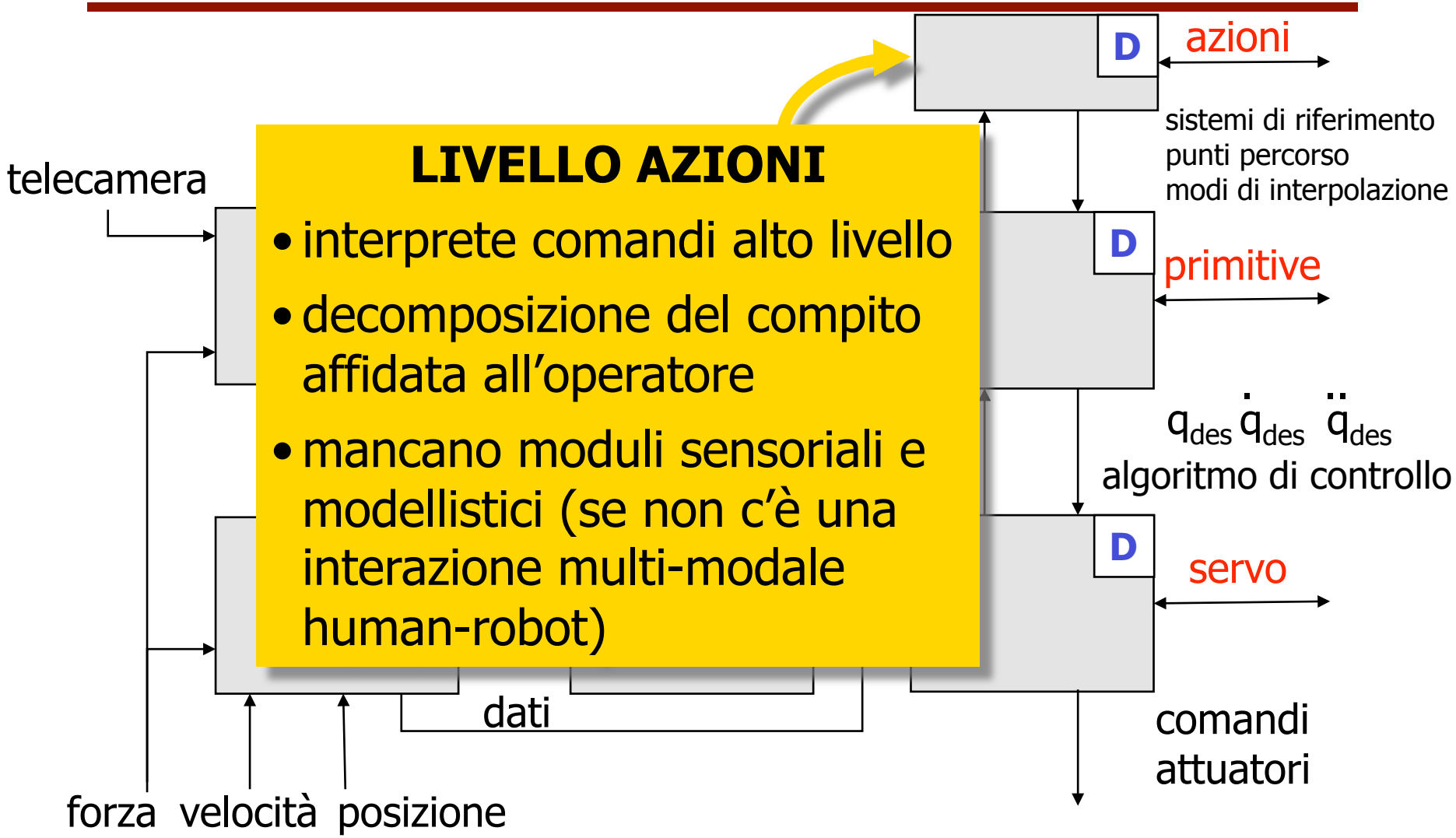
- Livello del compito: obiettivo del compito (specificato da operatore) analizzato e decomposto in azioni (basandosi su modelli di conoscenza di sistema robotico e ambiente)
- Livello delle azioni: comandi simbolici tradotti in sequenze di configurazioni intermedie
- Livello delle primitive: calcolo di traiettorie di riferimento per livello servo, scelta della strategia di controllo
- Livello dei servo: realizzazione algoritmi di controllo, calcolo comandi di attuazione per servomotori



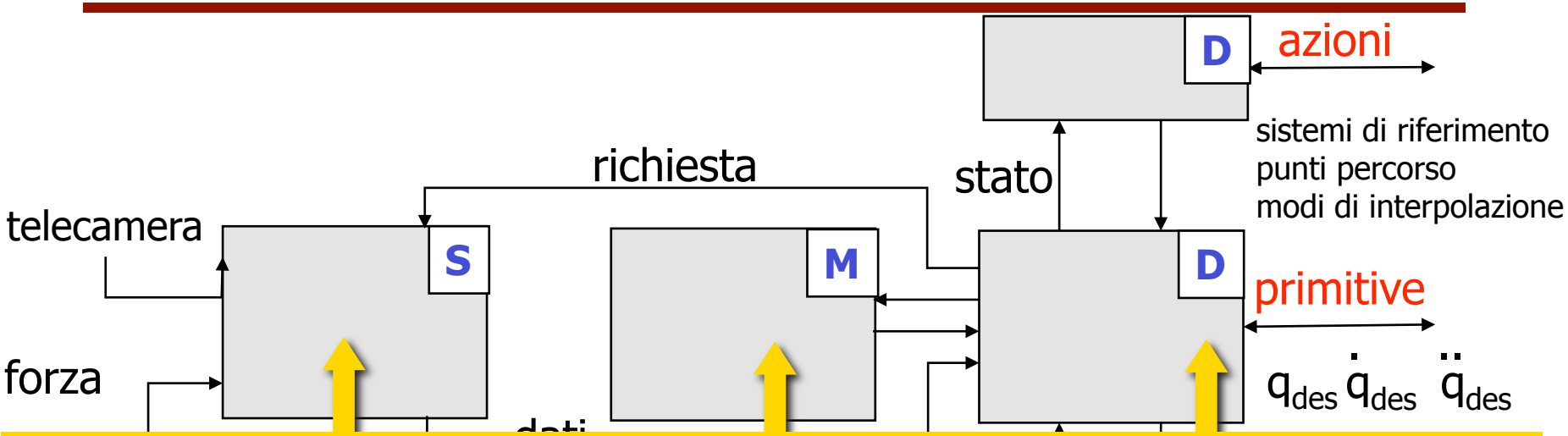
Un'architettura funzionale per robot industriali



Un'architettura funzionale per robot industriali



Un'architettura funzionale per robot industriali



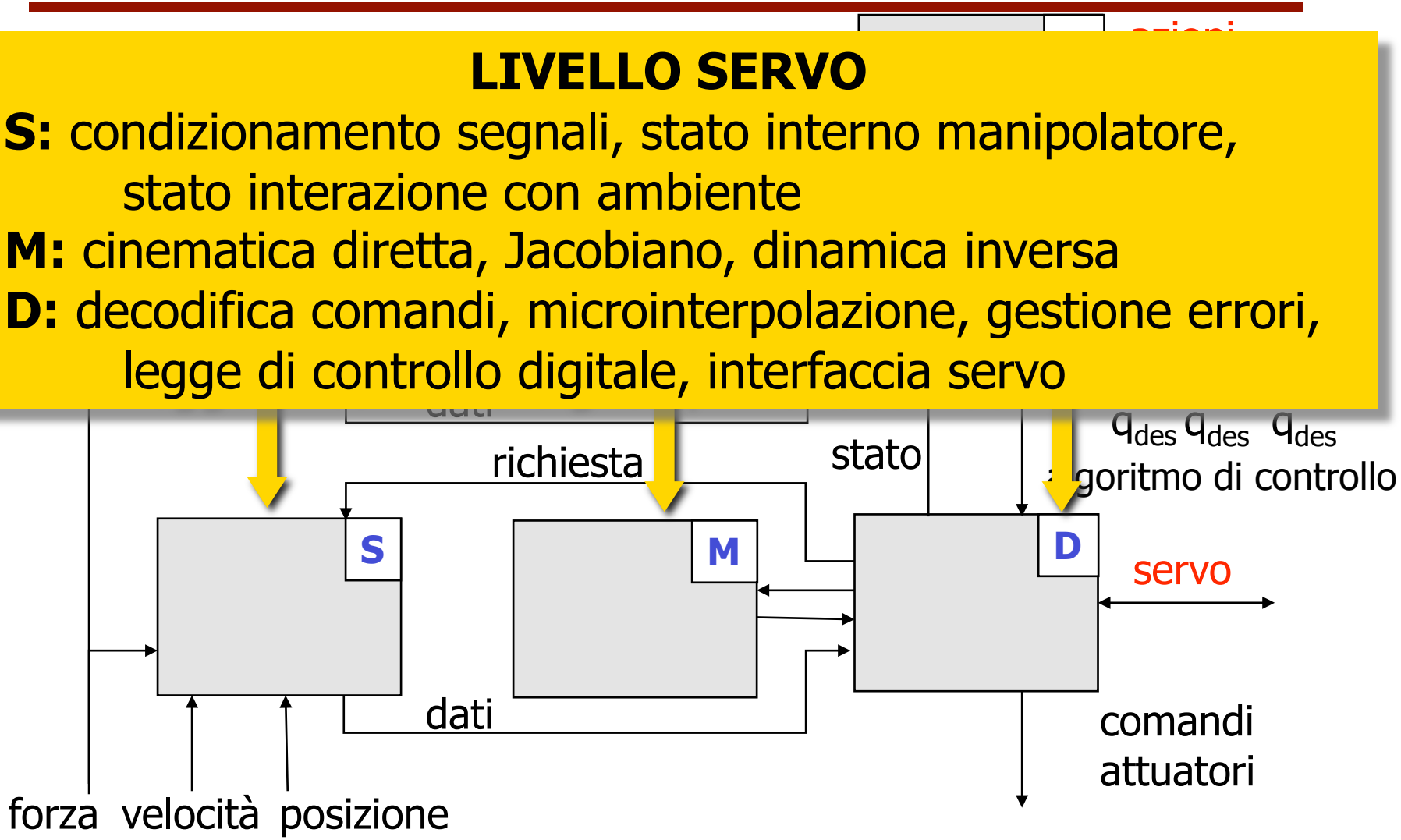
LIVELLO PRIMITIVE

- **S:** (solo per interazione attiva con ambiente) geometria dell'ambiente, stato interazione
- **M:** cinematica diretta e inversa, modelli dinamici
- **D:** decodifica comandi, generazione percorso, interpolazione traiettoria, inversione cinematica, analisi stato servo, gestione emergenze

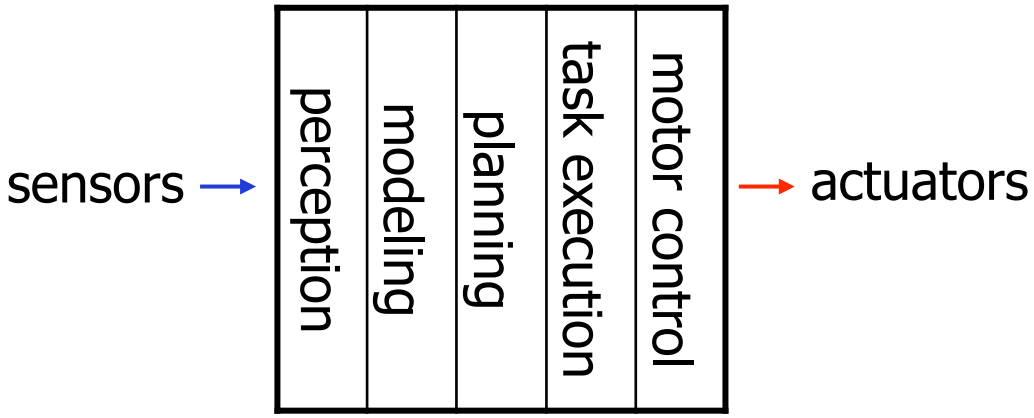
Un'architettura funzionale per robot industriali

LIVELLO SERVO

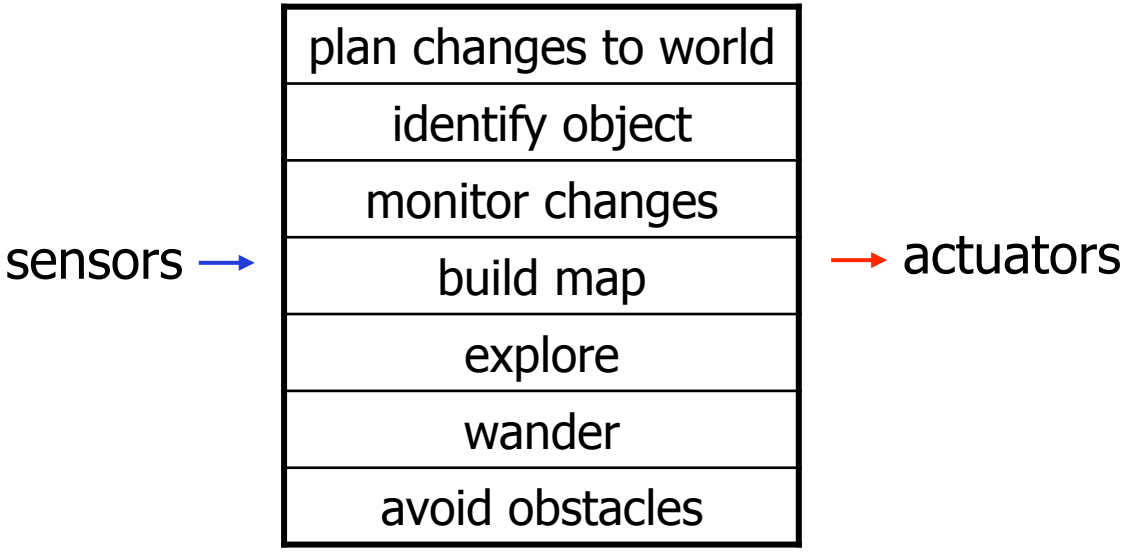
- **S:** condizionamento segnali, stato interno manipolatore, stato interazione con ambiente
- **M:** cinematica diretta, Jacobiano, dinamica inversa
- **D:** decodifica comandi, microinterpolazione, gestione errori, legge di controllo digitale, interfaccia servo



Interazione tra i moduli

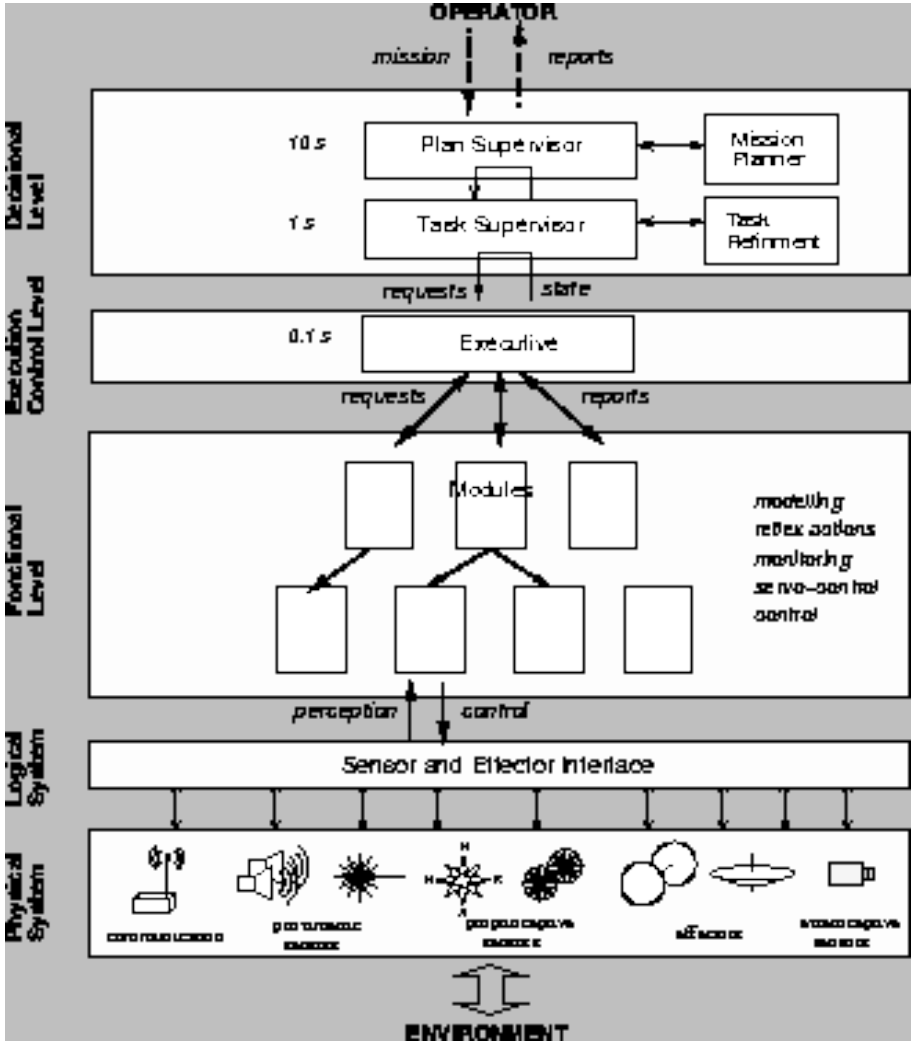


attivazione
orizzontale
(sequenziale)



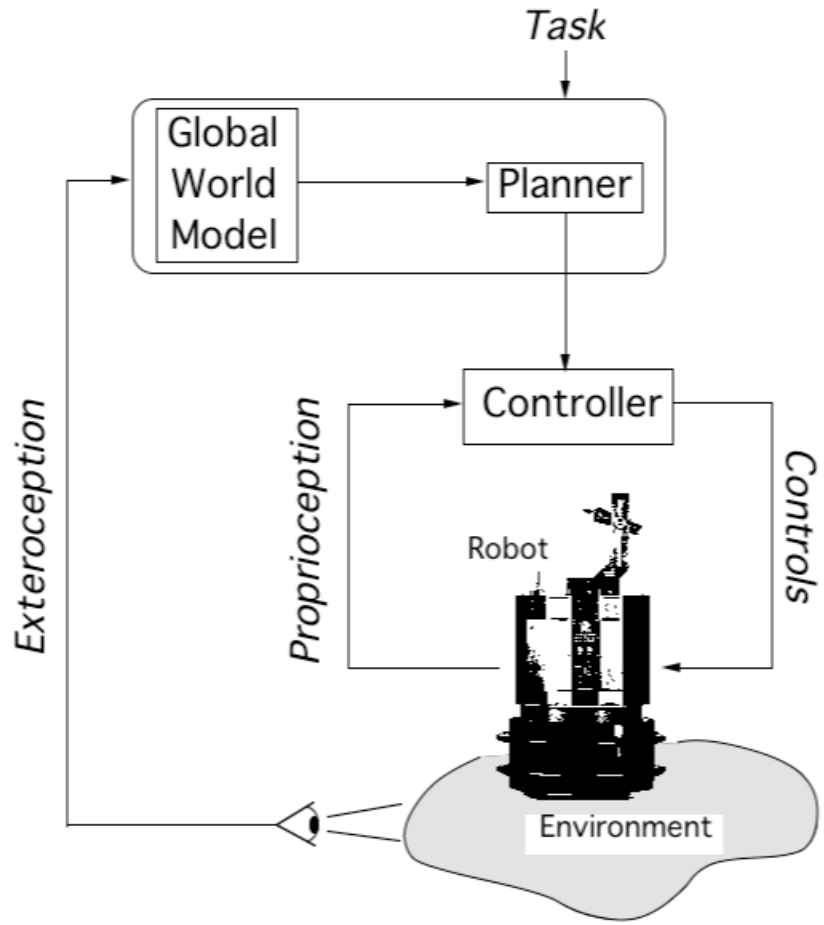
attivazione
verticale a richiesta
(subsumption)

Architettura del LAAS



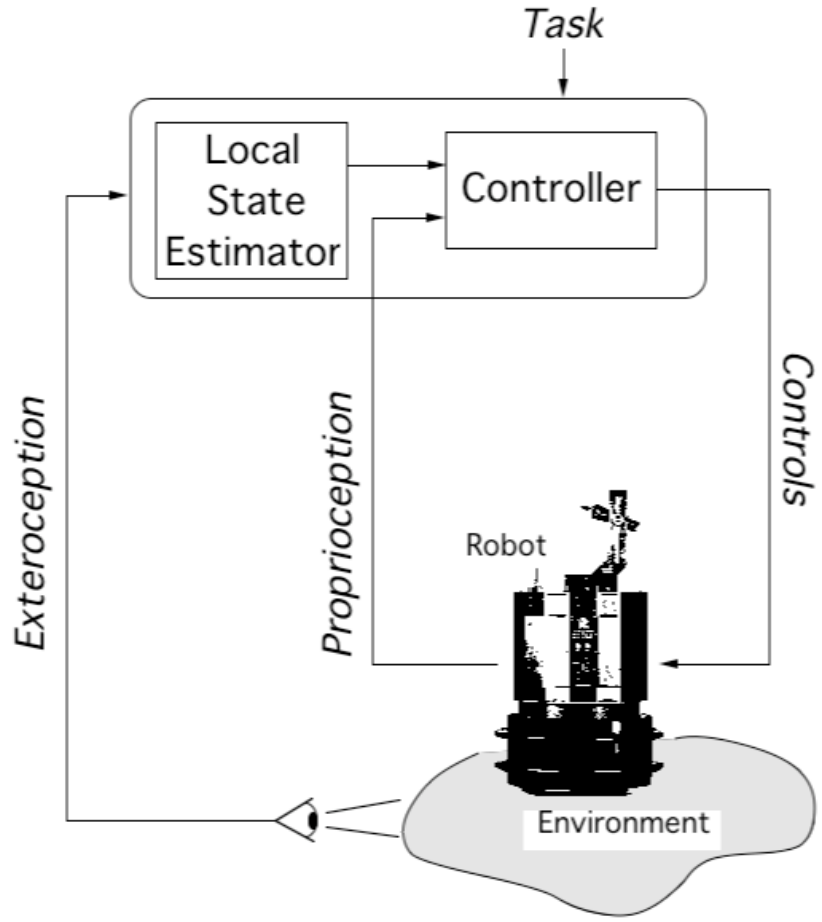
- esempio alternativo del LAAS/CNRS di Toulouse
- livello decisionale
- livello di sincronizzazione della esecuzione
- livello funzionale (moduli)
- livello con logica di interfacciamento
- livello fisico dei dispositivi

Sviluppo delle architetture - 1



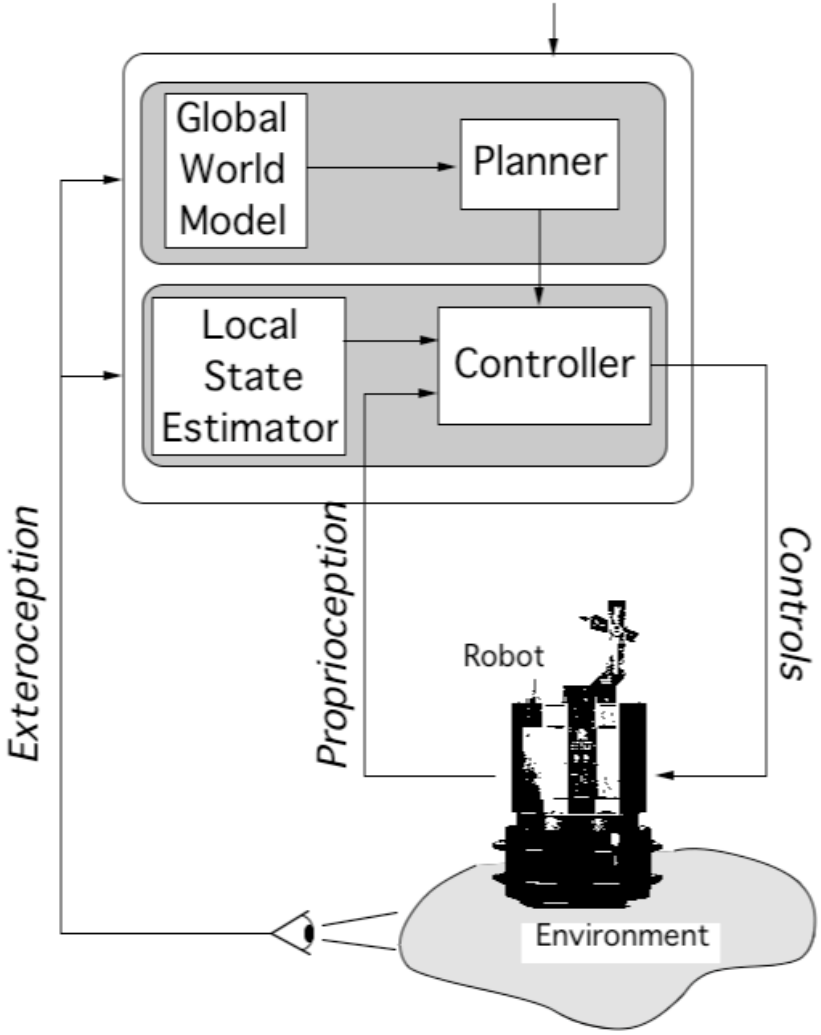
- esemplificato per la navigazione di un robot mobile su ruote
- sistema gerarchico
  - localizzazione iniziale
  - pianificazione off-line
  - controllo del moto on-line
  - possibilità eventuale di acquisizione/aggiornamento del modello dell'ambiente (con scala temporale lenta)

Sviluppo delle architetture - 2



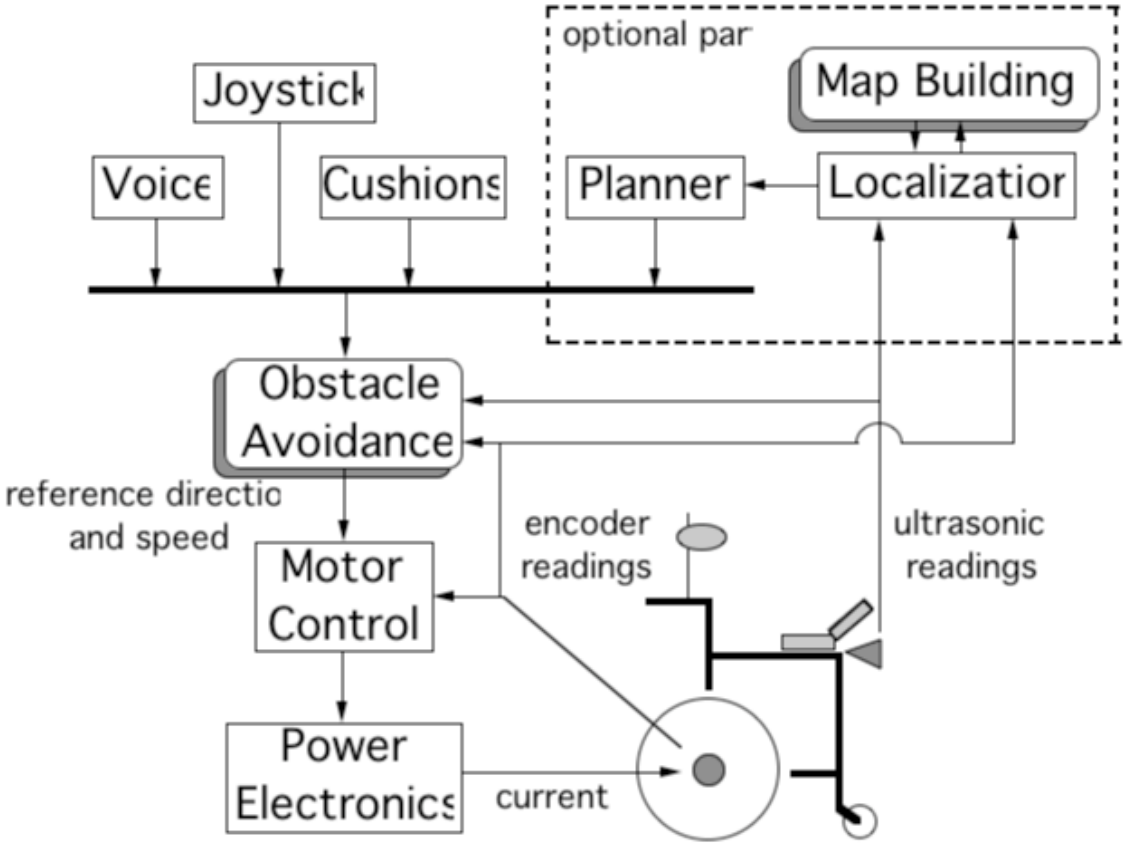
- sistema reattivo puro
  - stima in linea dell'ambiente locale (incognito)
  - obiettivo globale di posizionamento (goal)
  - reazione locale per avere "obstacle avoidance" + guida globale verso il goal

Sviluppo delle architetture - 3

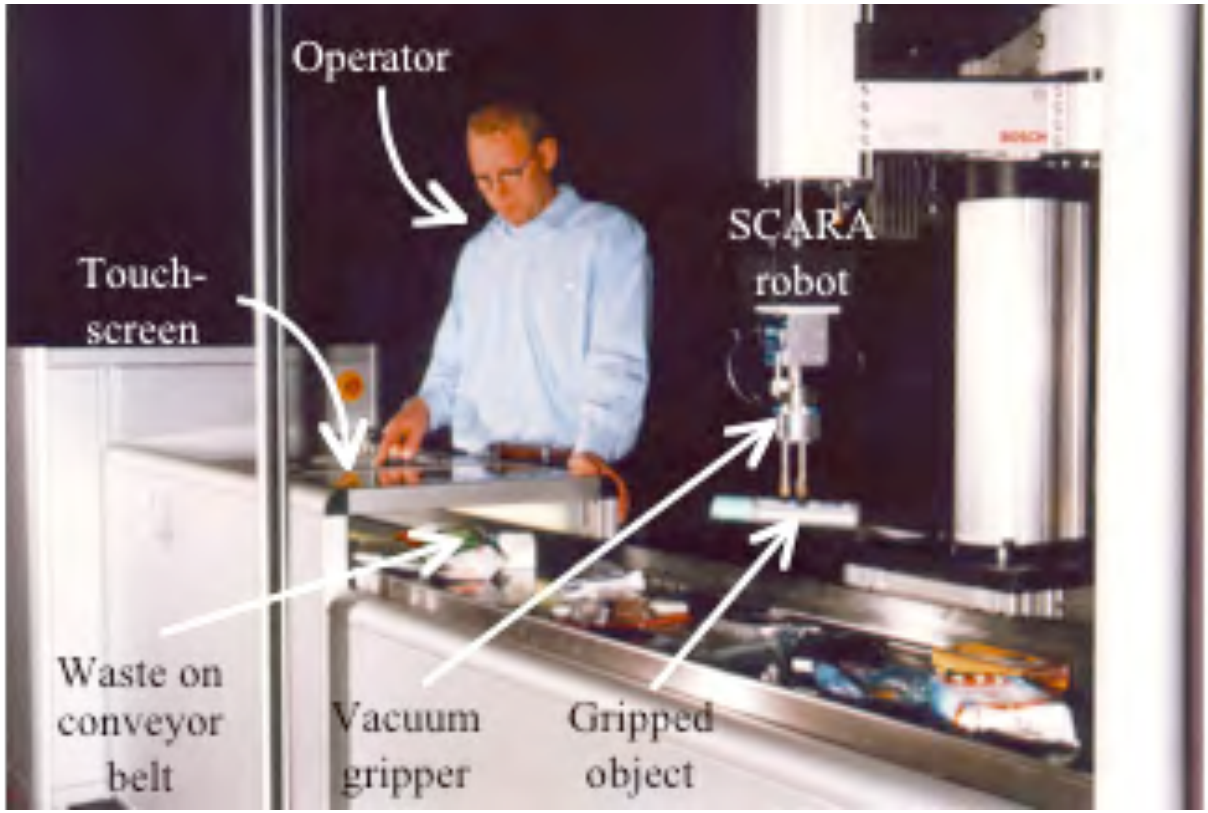


- sistema ibrido
  - SLAM = simultaneous localization and mapping
  - navigazione/esplorazione sul modello corrente
  - fusione di dati sensoriali
  - controllo del moto on-line

“Wheelchair” attuata e sensorizzata

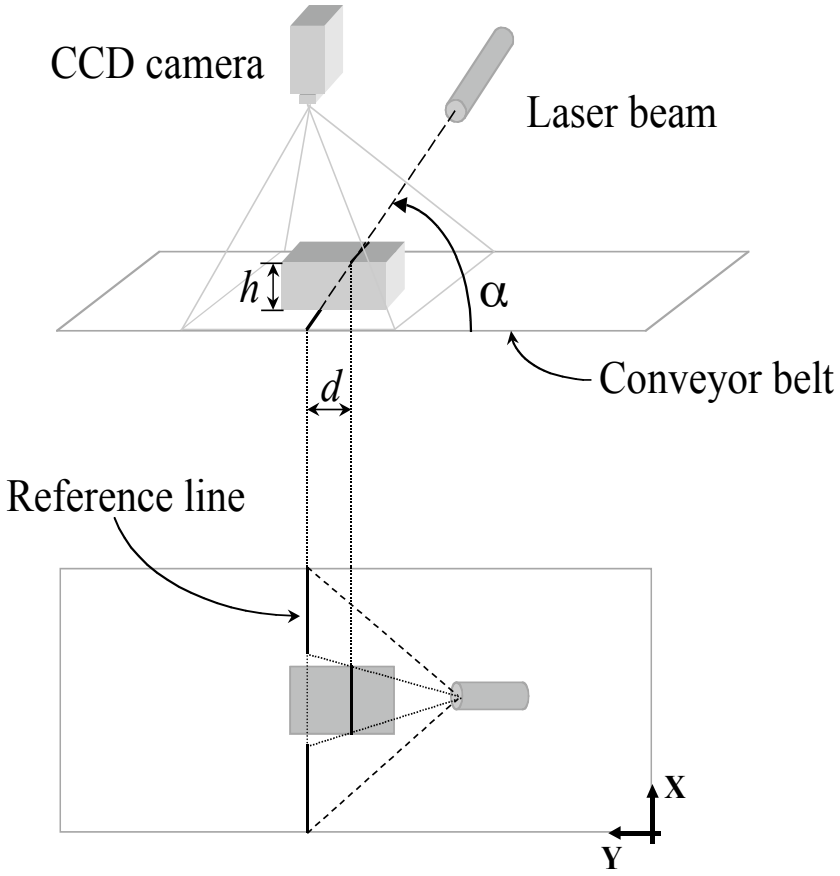


Cella per selezione di rifiuti domestici



versione semi-automatica
Fraunhofer IPA, Stoccarda

Modulo sensoriale
(nella versione automatica)

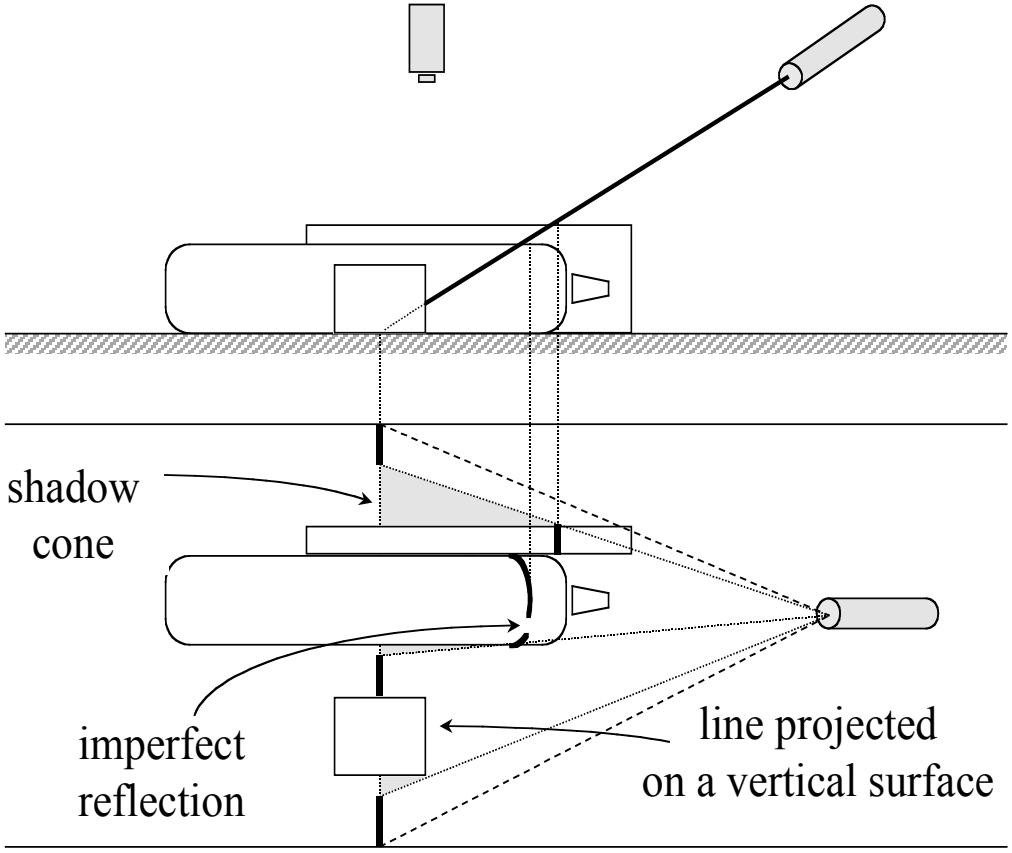


principio di funzionamento
del sensore a luce strutturata

operatore
+
touch-screen
sostituito da
visione a luce strutturata
+
sistema di localizzazione
neuro-fuzzy

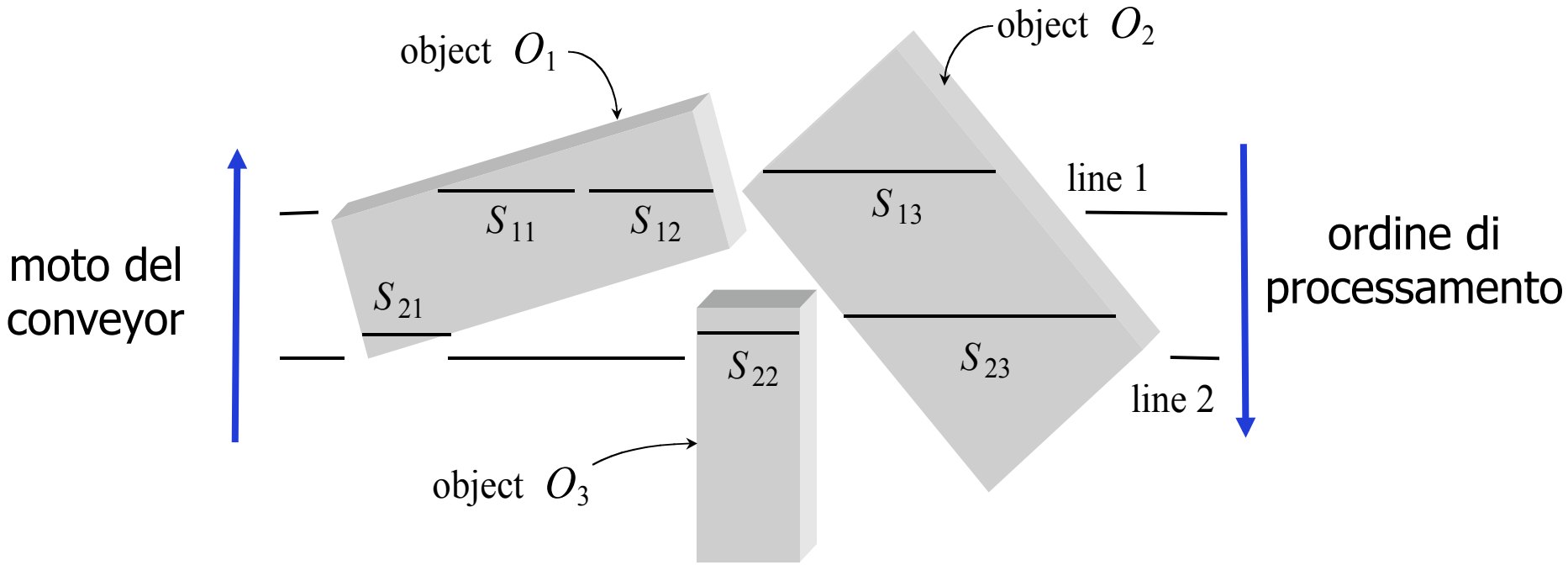
lavori della
Dr.ssa Raffaella Mattone

Interpretazione dati sensoriali



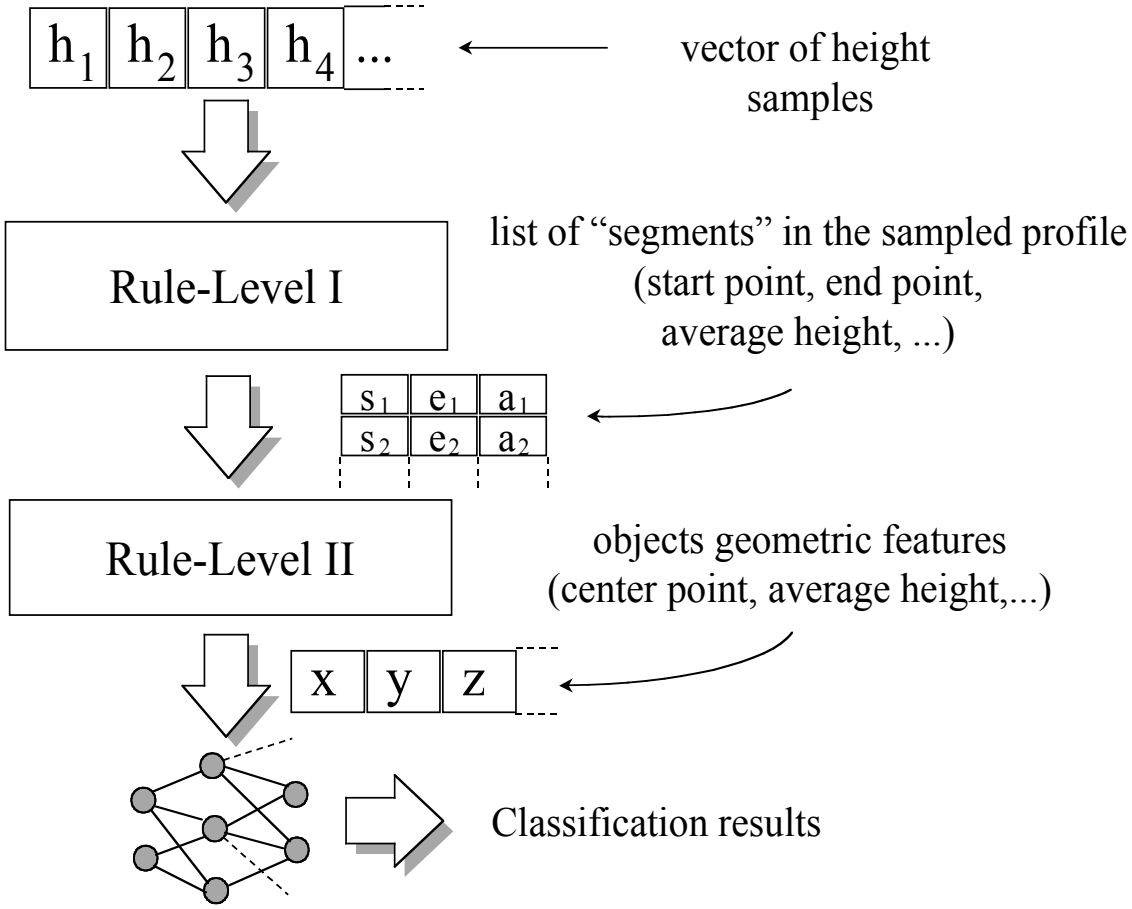
possibili cause di mancanza di informazione su una singola linea

Interpretazione dati sensoriali



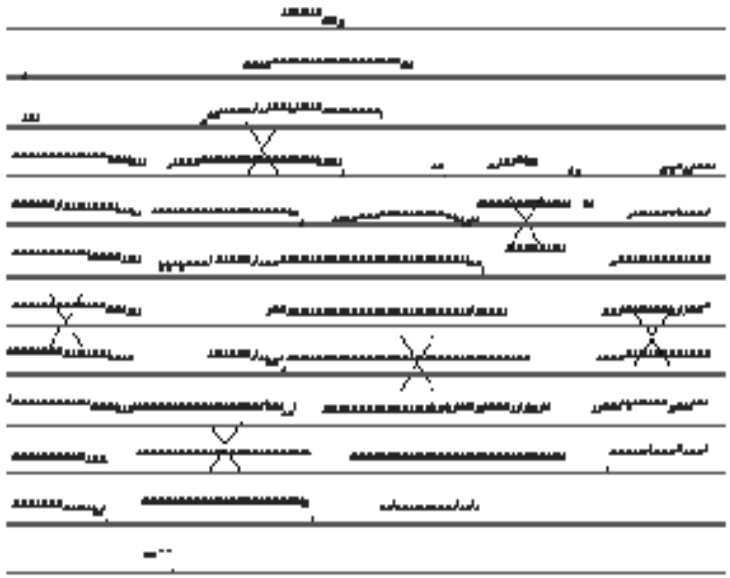
integrazione di dati ottenuti
ad istanti di campionamento successivi

Modulo decisionale



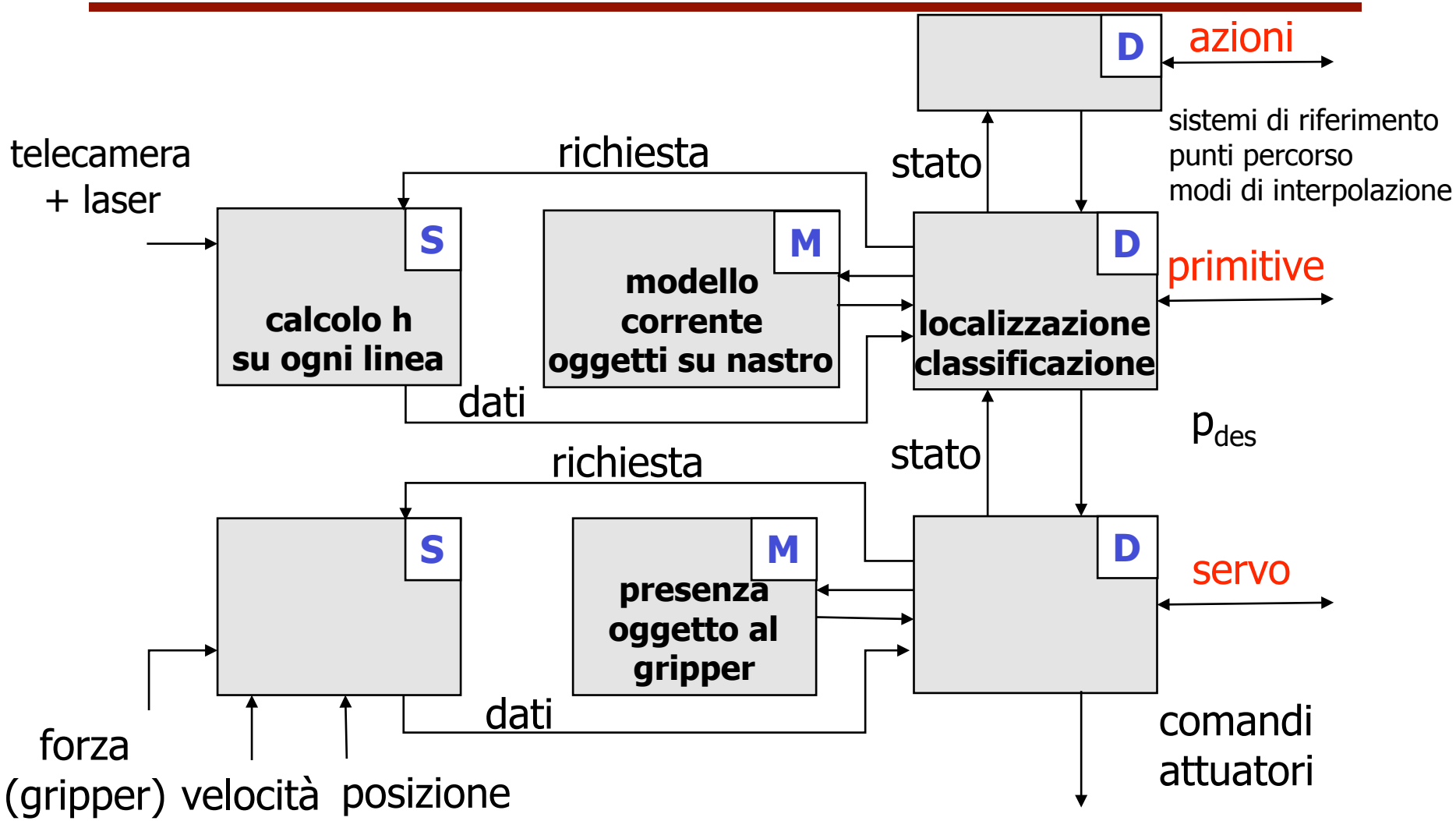
struttura del modulo di localizzazione e classificazione degli oggetti

Modulo modellistico

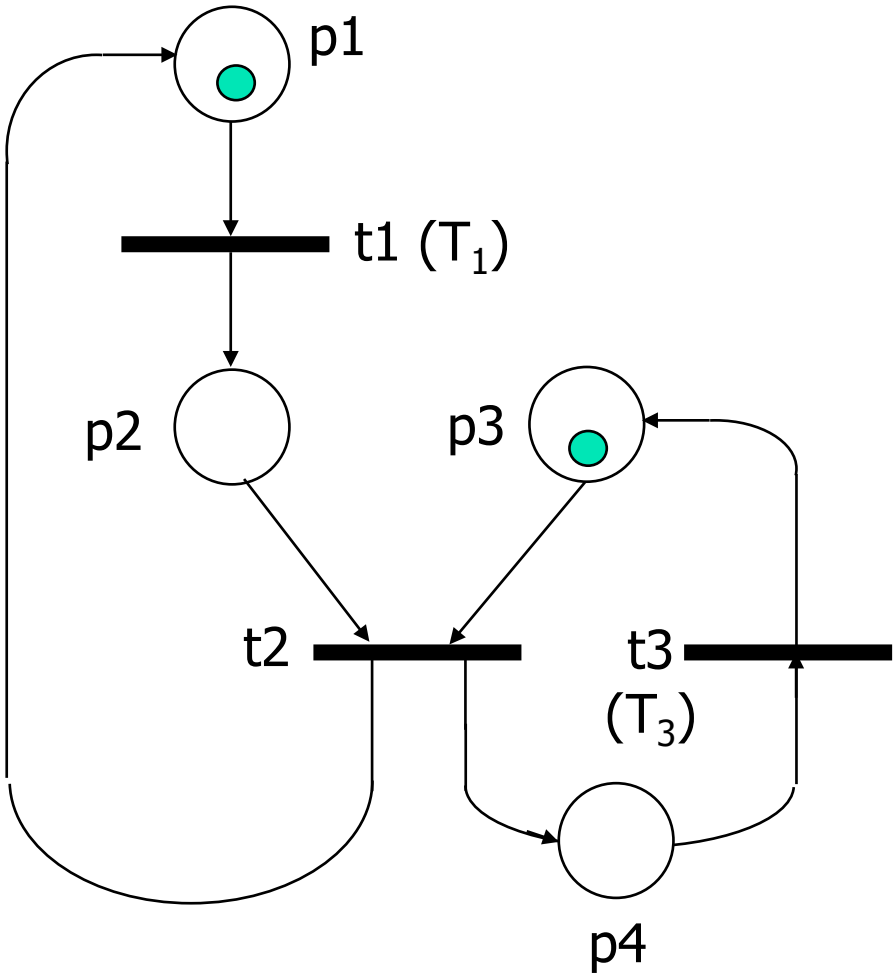


esempio di modello di oggetti sul nastro trasportatore

Architettura funzionale cella IPA



Descrizione con diagrammi di flusso

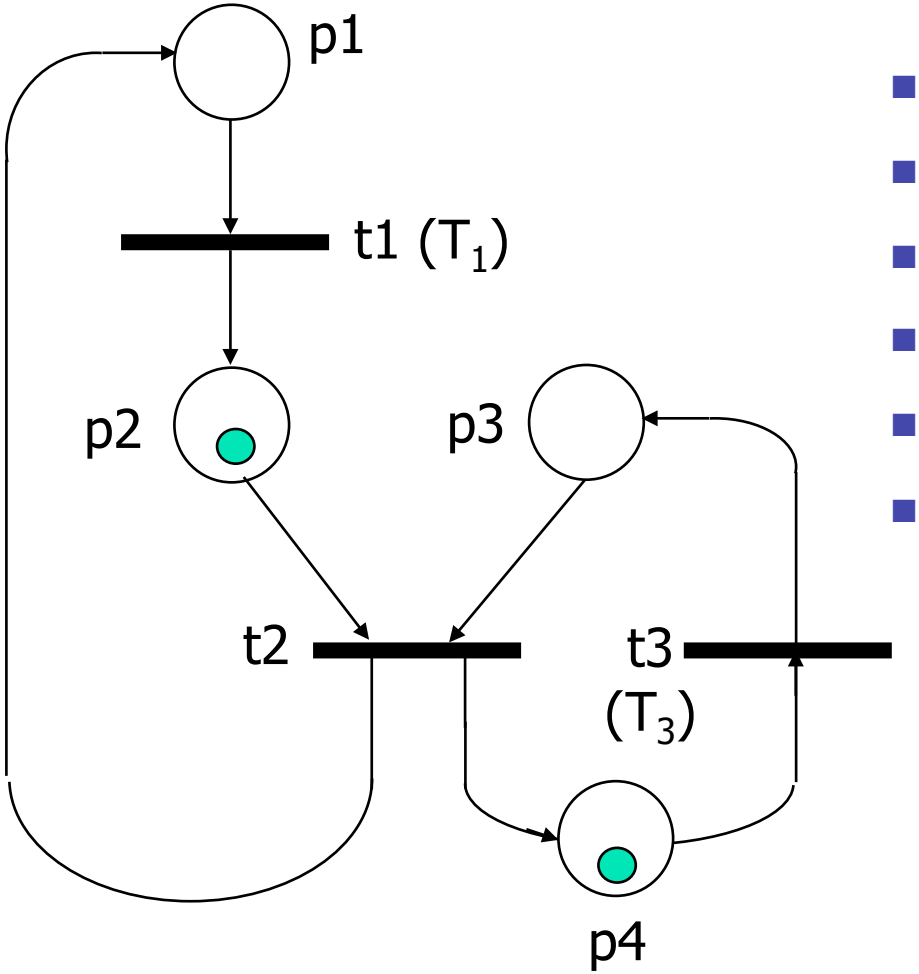


RETI DI PETRI

grafi orientati con due tipi di nodi

- posti (p1,…,p4)
  stati o blocchi funzionali: attivi se c'è un "token" nel posto (es. p1 e p3)
- transizioni (t1,…,t3)
  passaggi da uno stato a un altro attivati da eventi: presenza di tokens sufficienti (almeno uno) in tutti i posti in ingresso a transizione; eventualmente temporizzate (es. t1, t3)

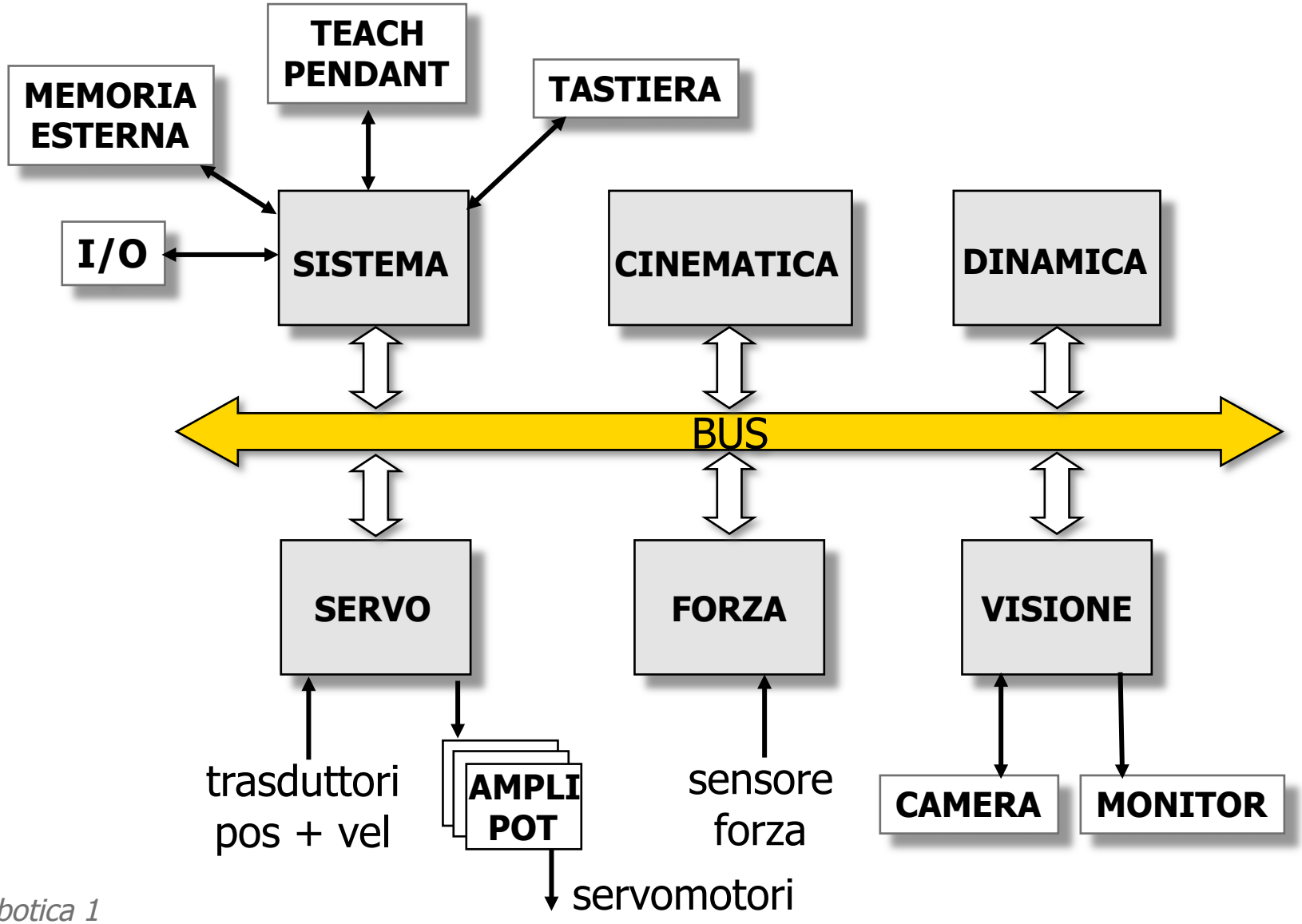
Diagramma di flusso cella IPA



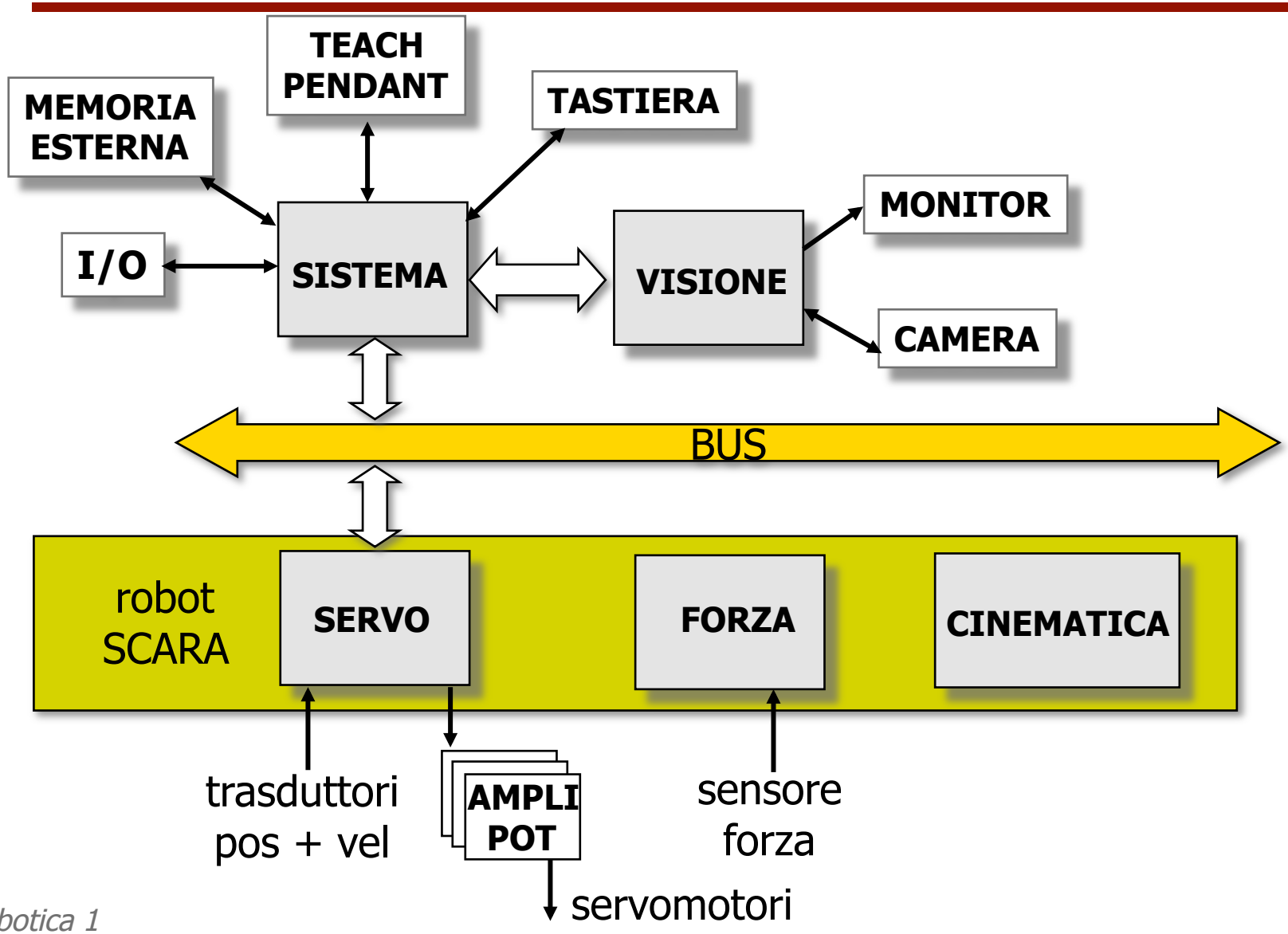
- p1: pick & place
- T_1: tempo di pick & place
- p2: robot ready
- p3: nuovo pezzo su conveyor
- p4: attesa nuovo pezzo
- T_3 (variabile aleatoria): intervallo di tempo tra due pezzi successivi

marcatura/condizione iniziale: robot pronto, attesa nuovo pezzo

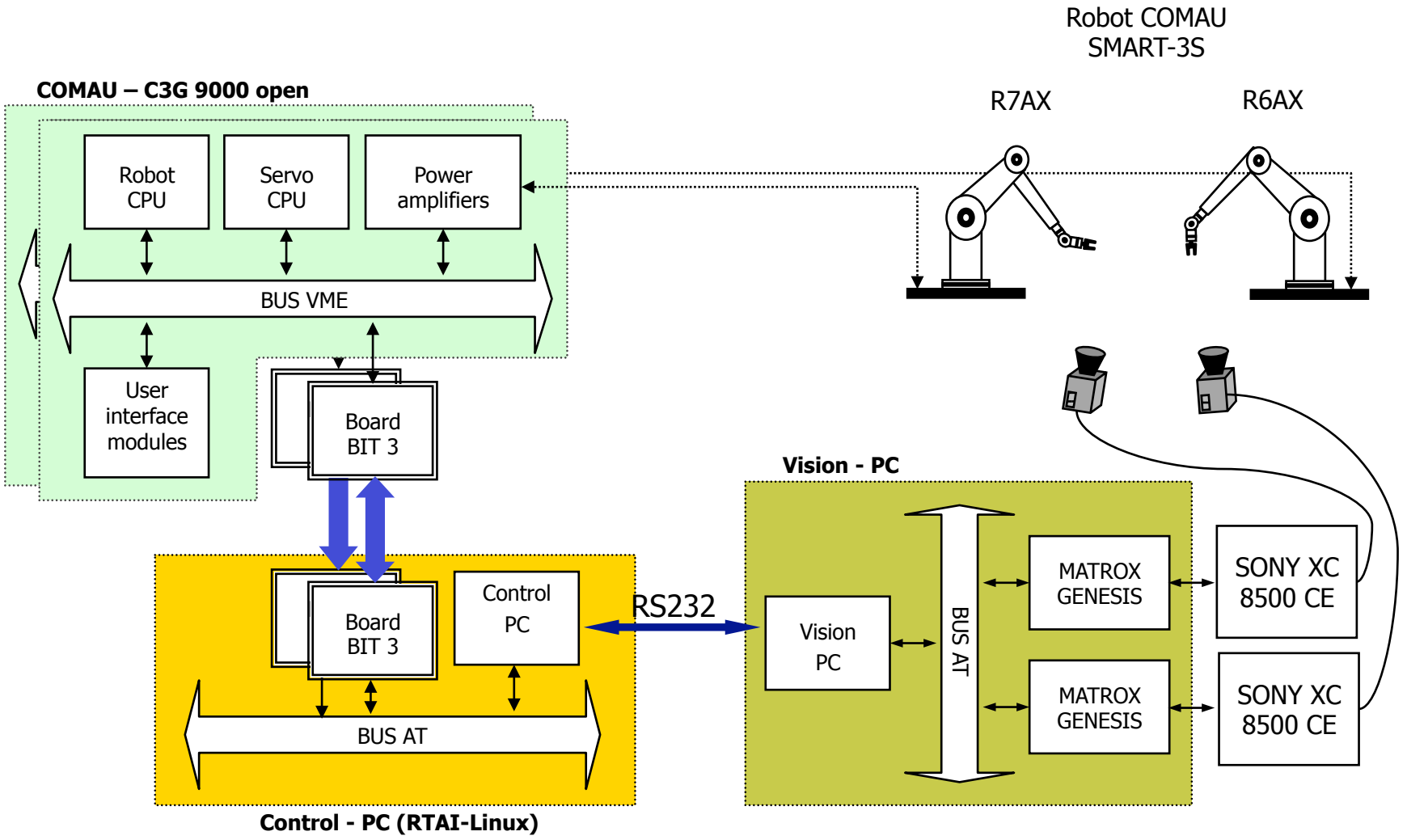
Architettura hardware



Architettura hardware cella IPA



Architettura hardware con visione



Ambienti di programmazione per robot

- sistema operativo in tempo reale
- modellazione del mondo
- controllo del moto
- lettura dati sensoriali
- interattività con sistema fisico
- rilevazione errori
- ripristino situazioni operative corrette
- struttura specifica del linguaggio

ambiente condizionato dal livello dell'architettura funzionale a cui è consentito l'accesso dell'operatore

Programmazione per insegnamento

- linguaggi di "prima generazione"
- programmazione ad insegnamento diretto (*teaching-by-showing*)
  - operatore guida robot (manualmente o tramite teach box) lungo percorso desiderato (off-line!)
  - posizioni di giunto campionate, interpolate e memorizzate per essere ripetute in seguito (accesso a livello di primitive)
  - generazione automatica del codice: non sono richieste capacità particolari di programmazione
- applicazioni: saldatura a punti, verniciatura, pallettizzazione
- esempi di linguaggi: T3 (Milacron), FUNKY (IBM)

Programmazione orientata al robot

- linguaggi di "seconda generazione": programmazione strutturata con caratteristiche di linguaggio interpretativo (ambiente di programmazione interattivo)
- integrazione di funzioni tipiche di linguaggi ad alto livello (ad es., condizionamento logico e cicli di attesa)
  - sviluppando linguaggi strutturati ad hoc per robot (più frequente)
  - sviluppando librerie specifiche per linguaggi standard (auspicabile)
- accesso a livello di azioni
- applicazioni complesse dove il robot interagisce con altre macchine all'interno di una cella di lavoro
- Esempi di linguaggi: VAL II (Unimation), AML (IBM), PDL 2 (Comau), KRL (Kuka)

Linguaggio KRL (Kuka)

- set di istruzioni

Variables and declarations	
DECL	(>>> 10.4.1 "DECL" page 138)
ENUM	(>>> 10.4.2 "ENUM" page 140)
IMPORT ... IS	(>>> 10.4.3 "IMPORT ... IS" page 141)
STRUC	(>>> 10.4.4 "STRUC" page 141)

Motion programming	
CIRC	(>>> 10.5.1 "CIRC" page 143)
CIRC_REL	(>>> 10.5.2 "CIRC_REL" page 144)
LIN	(>>> 10.5.3 "LIN" page 146)
LIN_REL	(>>> 10.5.4 "LIN_REL" page 146)
PTP	(>>> 10.5.5 "PTP" page 148)
PTP_REL	(>>> 10.5.6 "PTP_REL" page 148)

Program execution control	
CONTINUE	(>>> 10.6.1 "CONTINUE" page 150)
EXIT	(>>> 10.6.2 "EXIT" page 150)
FOR ... TO ... ENDFOR	(>>> 10.6.3 "FOR ... TO ... ENDFOR" page 150)
GOTO	(>>> 10.6.4 "GOTO" page 151)
HALT	(>>> 10.6.5 "HALT" page 152)
IF ... THEN ... ENDIF	(>>> 10.6.6 "IF ... THEN ... ENDIF" page 152)
LOOP ... ENDLOOP	(>>> 10.6.7 "LOOP ... ENDLOOP" page 153)
REPEAT ... UNTIL	(>>> 10.6.8 "REPEAT ... UNTIL" page 153)
SWITCH ... CASE ... ENDSWITCH	(>>> 10.6.9 "SWITCH ... CASE ... ENDSWITCH" page 154)
WAIT ... FOR	(>>> 10.6.10 "WAIT FOR" page 155)
WAIT ... SEC	(>>> 10.6.11 "WAIT SEC" page 156)
WHILE ... ENDWHILE	(>>> 10.6.12 "WHILE ... ENDWHILE" page 156)

Inputs/outputs	
ANIN	(>>> 10.7.1 "ANIN" page 157)
ANOUT	(>>> 10.7.2 "ANOUT" page 158)
DIGIN	(>>> 10.7.3 "DIGIN" page 159)
PULSE	(>>> 10.7.4 "PULSE" page 160)
SIGNAL	(>>> 10.7.5 "SIGNAL" page 164)

Subprograms and functions	
RETURN	(>>> 10.8.1 "RETURN" page 165)

Interrupt programming	
BRAKE	(>>> 10.9.1 "BRAKE" page 166)
INTERRUPT	(>>> 10.9.2 "INTERRUPT" page 166)
INTERRUPT ... DECL ... WHEN ... DO	(>>> 10.9.3 "INTERRUPT ... DECL ... WHEN ... DO" page 167)
RESUME	(>>> 10.9.4 "RESUME" page 169)

Path-related switching actions (=Trigger)	
TRIGGER WHEN DISTANCE	(>>> 10.10.1 "TRIGGER WHEN DISTANCE" page 170)
TRIGGER WHEN PATH	(>>> 10.10.2 "TRIGGER WHEN PATH" page 173)

Communication
(>>> 10.11 "Communication" page 176)

System functions	
VARSTATE()	(>>> 10.12.1 "VARSTATE()" page 176)

Linguaggio KRL (Kuka)

- tipiche primitive di moto

moto PTP
(point-to-point,
lineare nei giunti)

moto LIN
(lineare nel
cartesiano)

moto CIRC
(circolare nel
cartesiano)

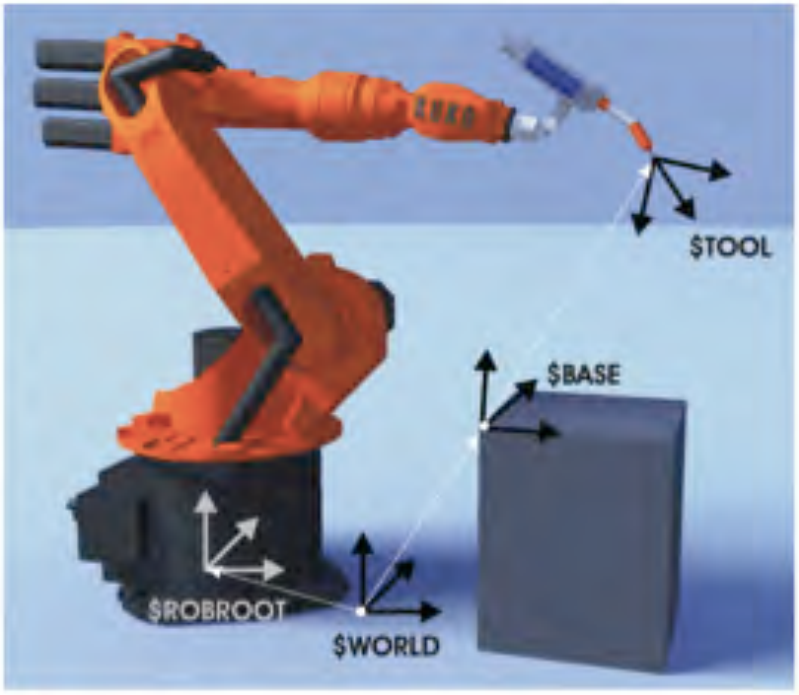
moto PTP
(lineare negli angoli)

orientamento
organo terminale

orientamento COST

Linguaggio KRL (Kuka)

- sistemi di coordinate (frames cartesiani) e jogging dei giunti



Linguaggio KRL (Kuka)

- teach pendant (interfaccia d'utente)

Fig. 4-1: Front view of KCP

1	Mode selector switch	10	Numeric keypad
2	Drives ON	11	Softkeys
3	Drives OFF / SSB GUI	12	Start backwards key
4	EMERGENCY STOP button	13	Start key
5	Space Mouse	14	STOP key
6	Right-hand status keys	15	Window selection key
7	Enter key	16	ESC key
8	Arrow keys	17	Left-hand status keys
9	Keypad	18	Menu keys

Programmazione orientata al compito

- linguaggi di "terza generazione" (di ricerca, non ancora disponibili sul mercato)
- simile a object-oriented programming
- compito specificato da istruzioni ad alto livello che realizzano azioni sulle parti presenti nella scena (sistemi esperti, intelligenza artificiale)
- accesso a livello del compito