Agentic AI

UNIVERSITY OF BRESCIA

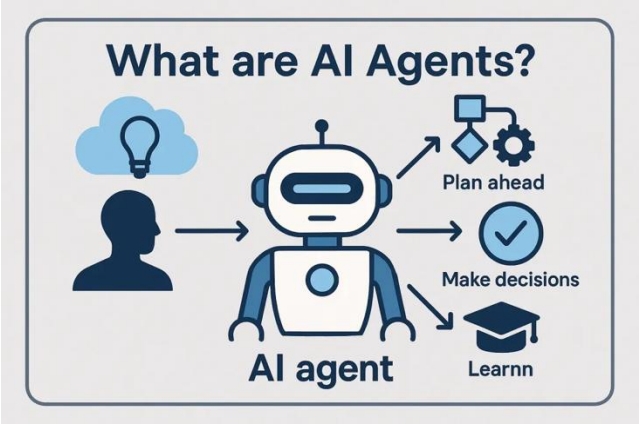
Intelligenza Artificiale

Mattia Chiari

Agente

Definizione

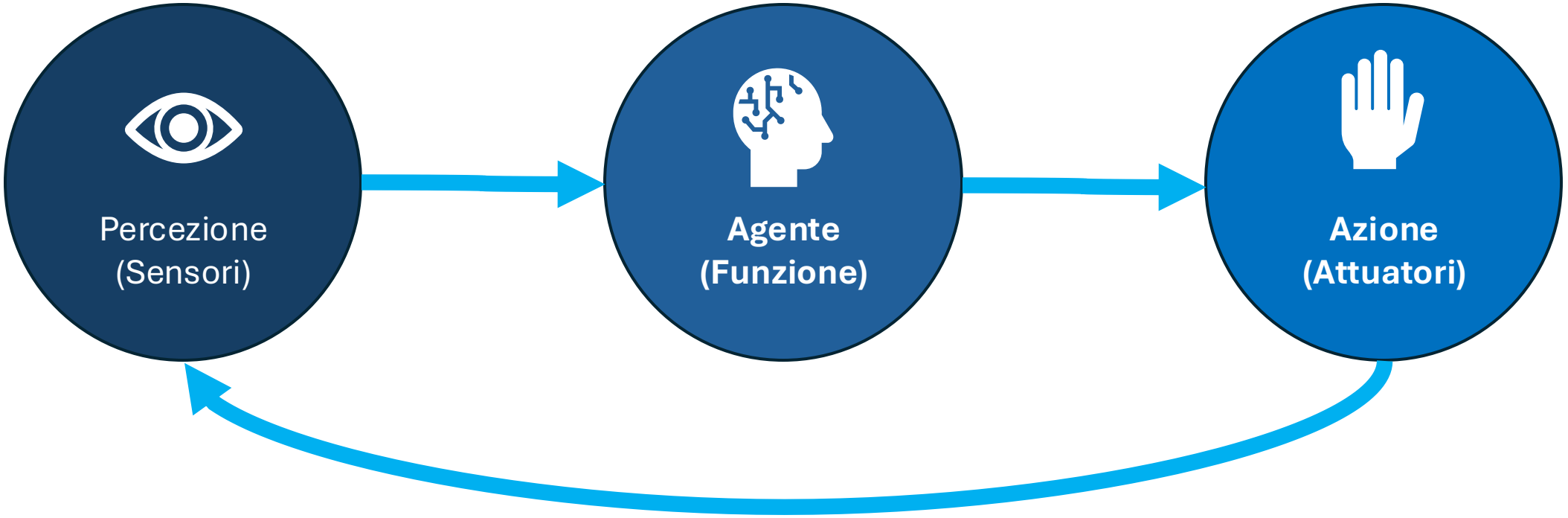
Un **agente** è un sistema che interagisce con un ambiente, percependone lo stato (o parte di esso) tramite sensori e producendo azioni su tale ambiente tramite attuatori.



Fonte: AWS

Operativamente, un agente può essere visto come una funzione che, a ogni sequenza di percezioni ricevute finora, associa l’azione successiva da eseguire.

Agente (2)



Agente Razionale

Definizione

Un **agente razionale** è un agente che, per ogni possibile sequenza di percezioni, sceglie l'azione che massimizza il valore atteso di una data misura di prestazione, sulla base delle informazioni disponibili e delle conoscenze di cui dispone.

Ambiente

Definizione

L’ambiente è il luogo fisico o virtuale in cui agisce a con cui interagisce l’agente.

Fonte: Mecalux

Fonte: Minecraft

Caratteristiche dell'Ambiente

Osservabile	(Parzialmente) Non-Osservabile
Deterministico	Stocastico
Episodico	Sequenziale
Statico	Dinamico

Tipologie di Agente

- Agente a Riflesso Semplice
- Agente a Riflesso basati su Modello
- Agente basato sul Goal
- Agente basato sull’Utilità
- Agente basato sull’Apprendimento

Agente a Riflesso Semplice

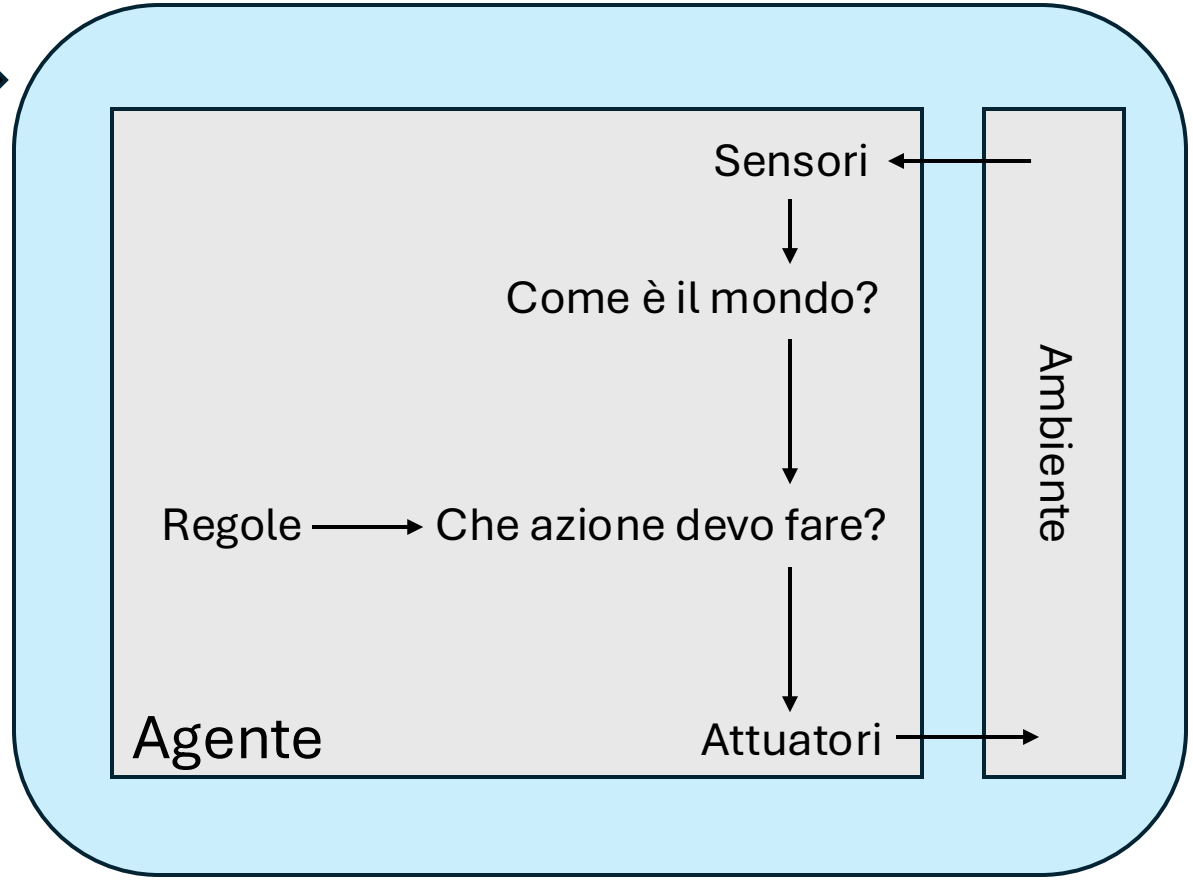
Agente a Riflesso Semplice

Agente a Riflesso basati su Modello

Agente basato sul Goal

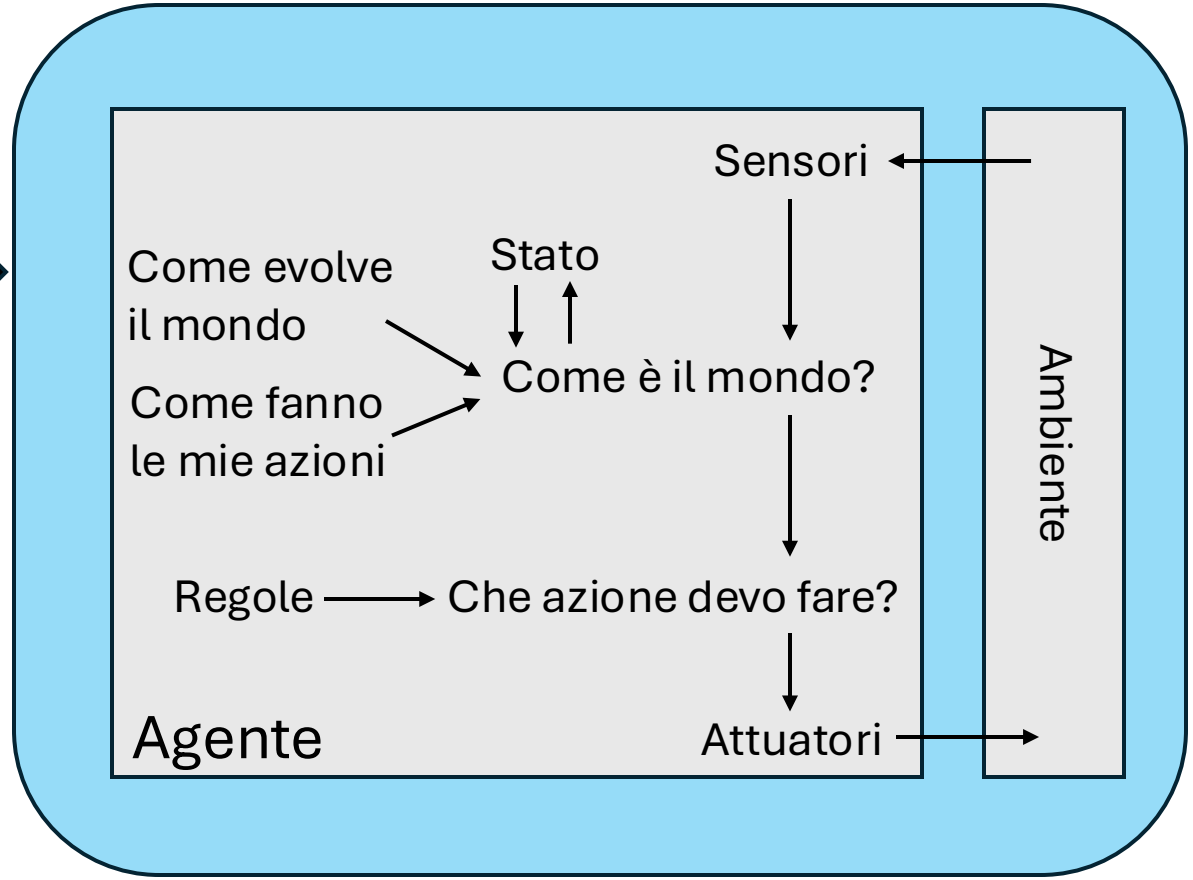
Agente basato sull'Utilità

Agente basato sull'Apprendimento



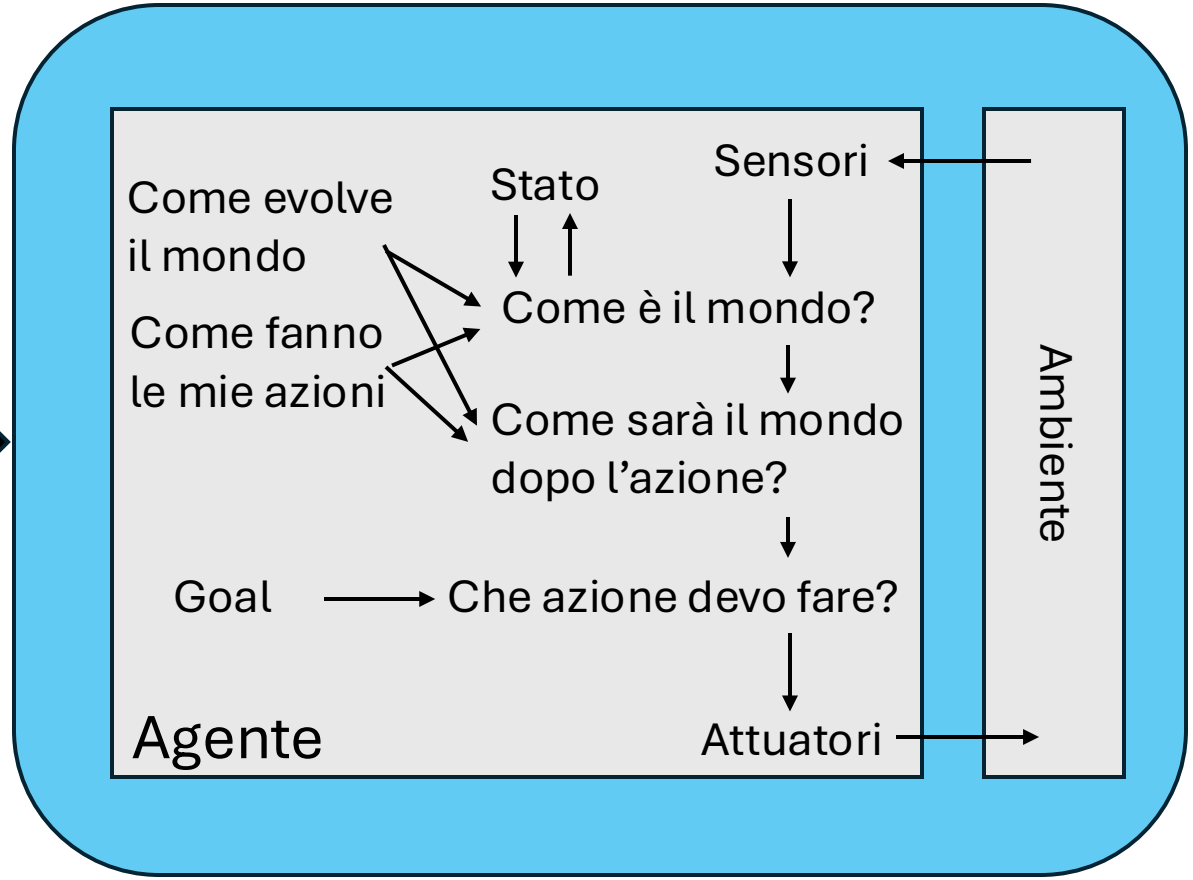
Agente a Riflesso basato su Modello

- Agente a Riflesso Semplice
- **Agente a Riflesso basati su Modello**
- Agente basato sul Goal
- Agente basato sull'Utilità
- Agente basato sull'Apprendimento



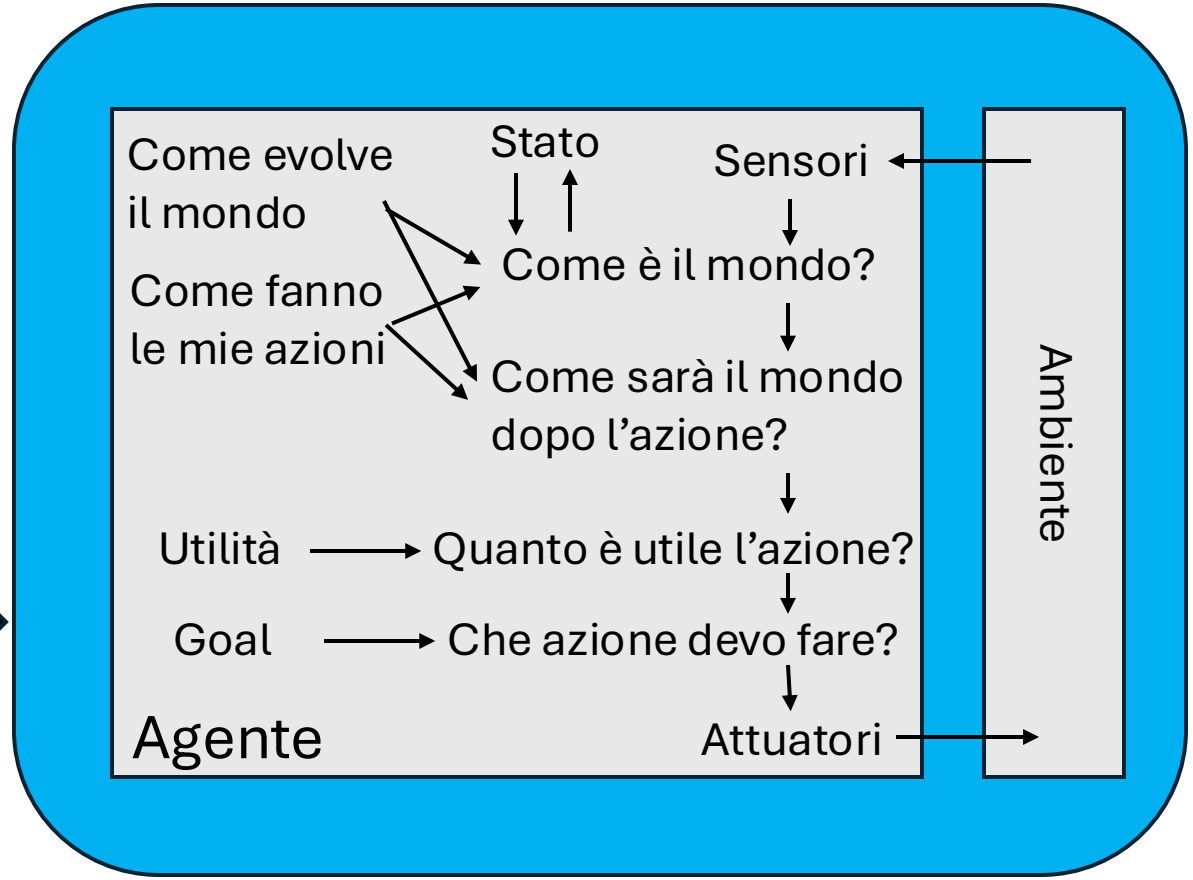
Agente basato sul Goal

- Agente a Riflesso Semplice
- Agente a Riflesso basati su Modello
- **Agente basato sul Goal**
- Agente basato sull'Utilità
- Agente basato sull'Apprendimento



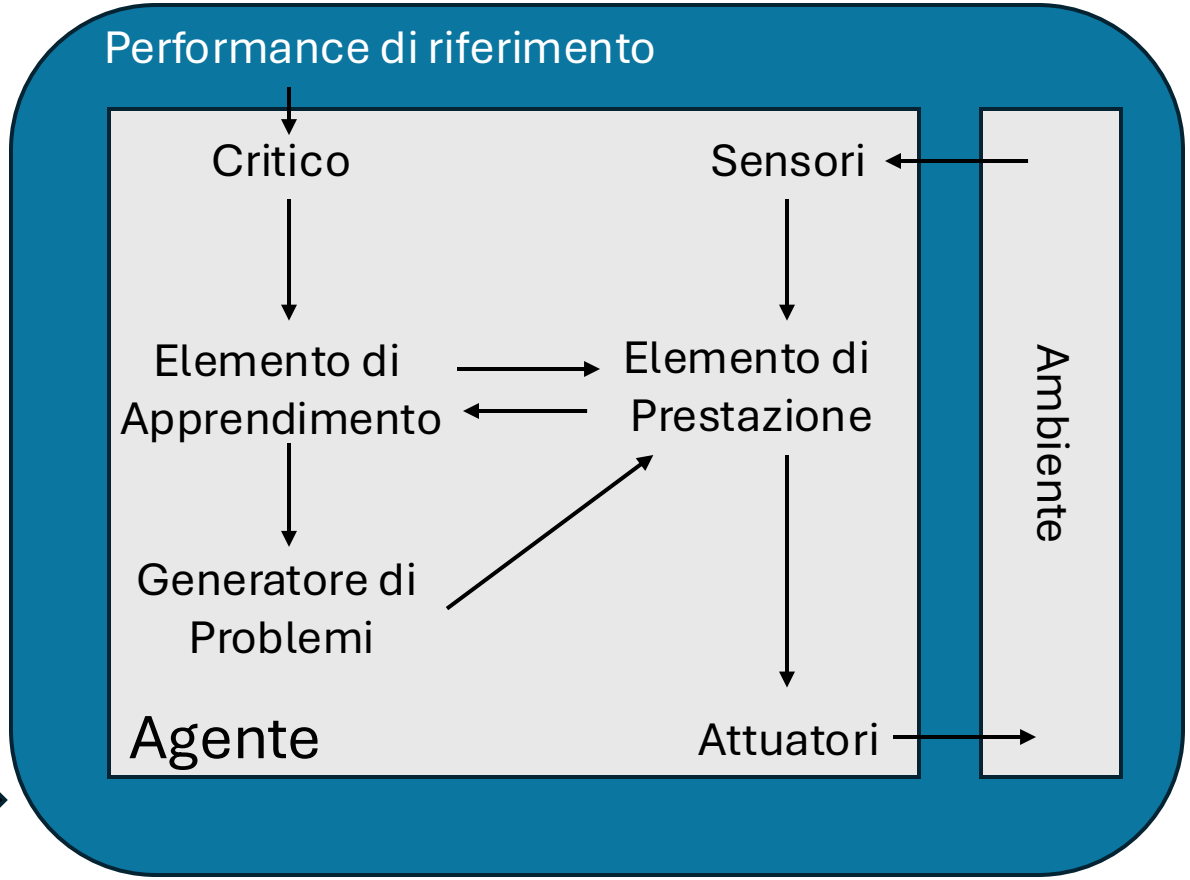
Agente basato sull'Utilità

- Agente a Riflesso Semplice
- Agente a Riflesso basati su Modello
- Agente basato sul Goal
- **Agente basato sull'Utilità**
- Agente basato sull'Apprendimento



Agente basato sull'Apprendimento

- Agente a Riflesso Semplice
- Agente a Riflesso basati su Modello
- Agente basato sul Goal
- **Agente basato sull'Utilità**
- Agente basato sull'Apprendimento



Agentic AI

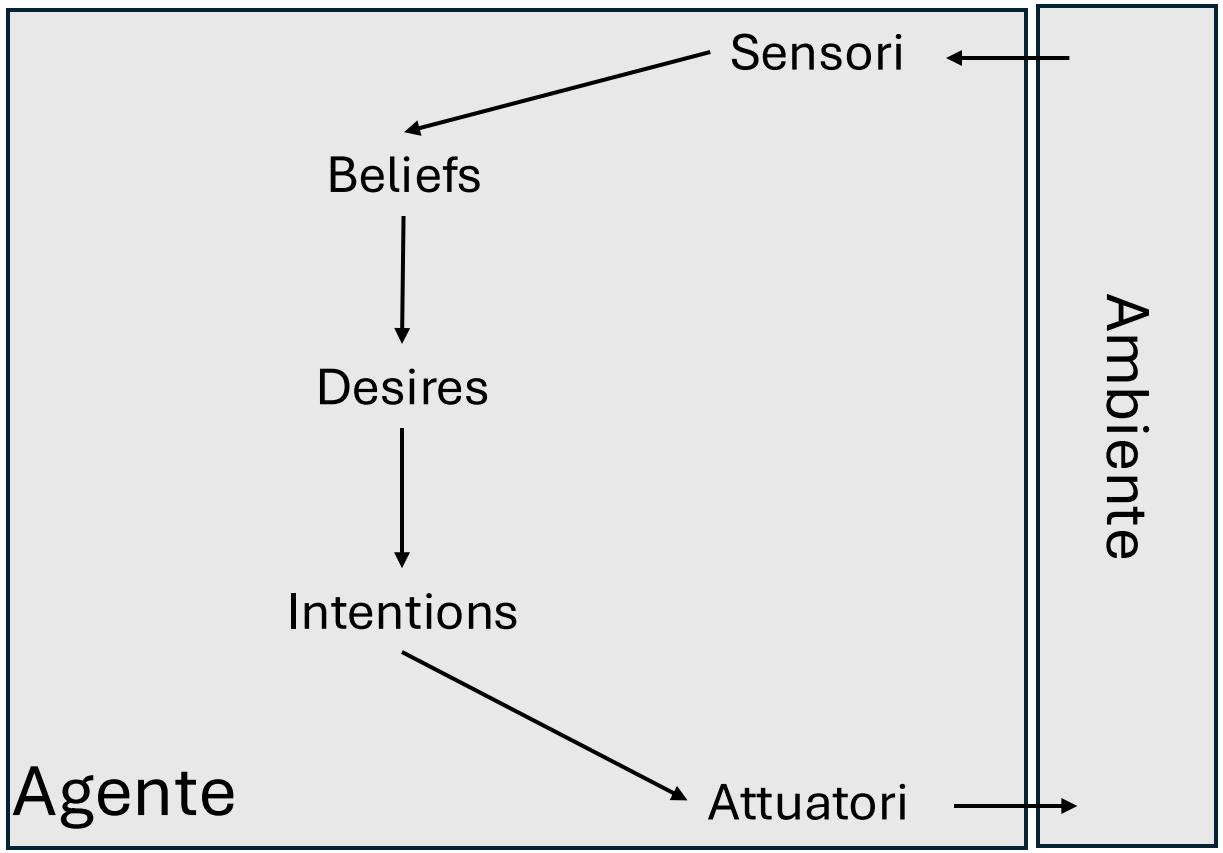
Definizione

Agentic AI indica una classe di sistemi di intelligenza artificiale progettati come **agenti autonomi**, capaci di perseguire obiettivi assegnati operando in modo iterativo su un ambiente.

Tali sistemi integrano capacità di **percezione, ragionamento, pianificazione e azione**, e possono utilizzare strumenti esterni, memoria persistente e, in molti casi, coordinarsi con altri agenti per raggiungere i propri obiettivi.

Dal punto di vista concettuale, Agentic AI rappresenta una continuazione dei modelli classici di **agenti razionali**, arricchiti dall'uso di LLM, strumenti esterni e architetture multi-agente.

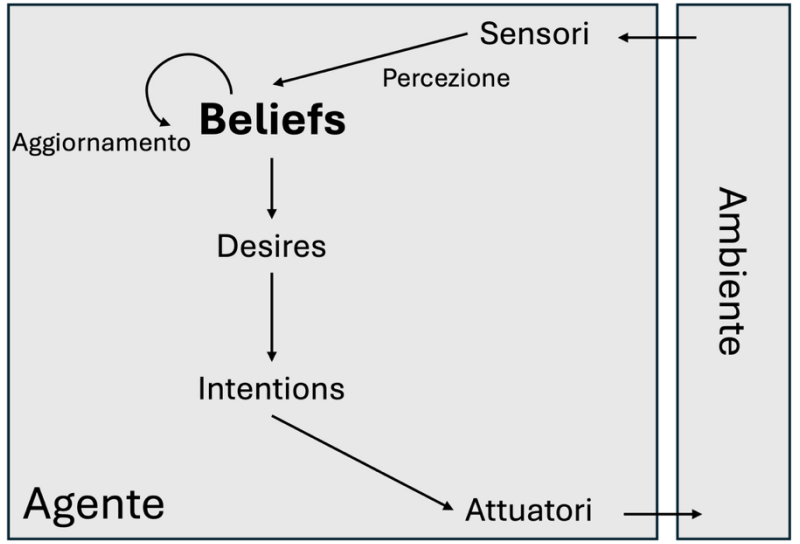
Agenti BDI


Sensori
Beliefs
Desires
Intentions
Agente
Attuatori
Ambiente

Agenti BDI: Beliefs

Definizione

Le **beliefs** rappresentano lo **stato informativo dell'agente**, cioè ciò che l'agente ritiene vero sul mondo, su se stesso e sugli altri agenti.



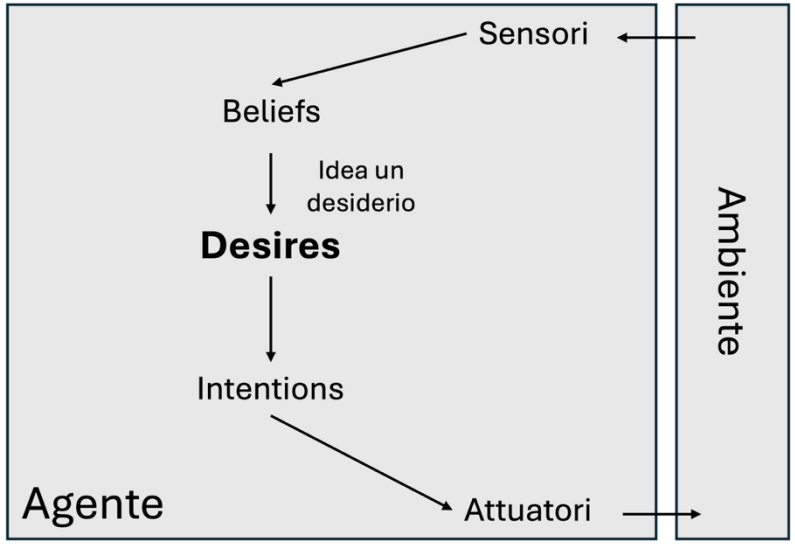
Possono includere **fatti e regole di inferenza**, che permettono di derivare nuove credenze tramite.

Il termine *credenze* è usato invece di *conoscenza* per sottolineare che le informazioni dell'agente **possono essere incomplete, incerte o non corrette**, e possono **cambiare nel tempo**.

Agenti BDI: Desires

Definizione

I **desires** rappresentano lo **stato motivazionale dell'agente**



Corrispondono a **obiettivi, condizioni o situazioni** che l'agente vorrebbe realizzare o rendere vere.

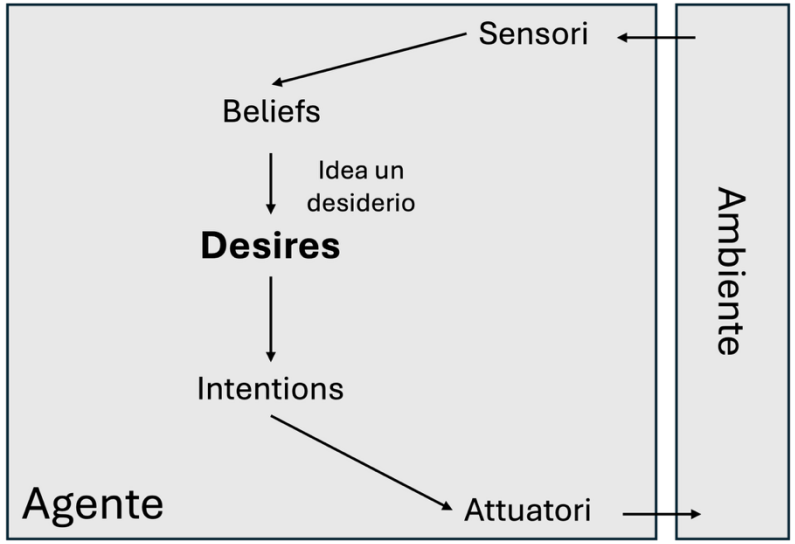
Esempi di desires possono essere:

- trovare il prezzo migliore,
- andare a una festa,
- diventare ricco.

Agenti BDI: Desires (2)

Definizione

Un **goal** è un **desire che l'agente ha deciso di perseguire attivamente**.



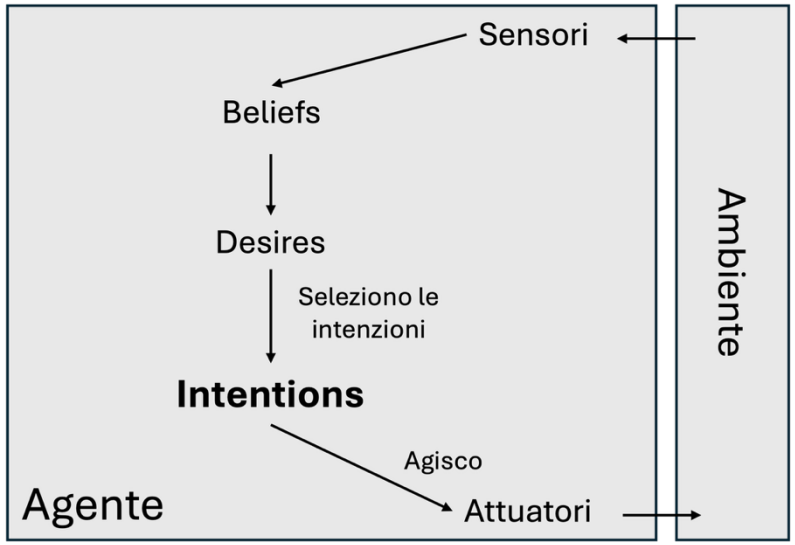
L'uso del termine *goal* introduce un vincolo aggiuntivo: l'insieme dei goal attivi deve essere **coerente**.

Ad esempio, un agente **non può avere contemporaneamente** il goal di andare a una festa e quello di restare a casa, anche se entrambe le opzioni possono essere desiderabili.

Agenti BDI: Intentions

Definizione

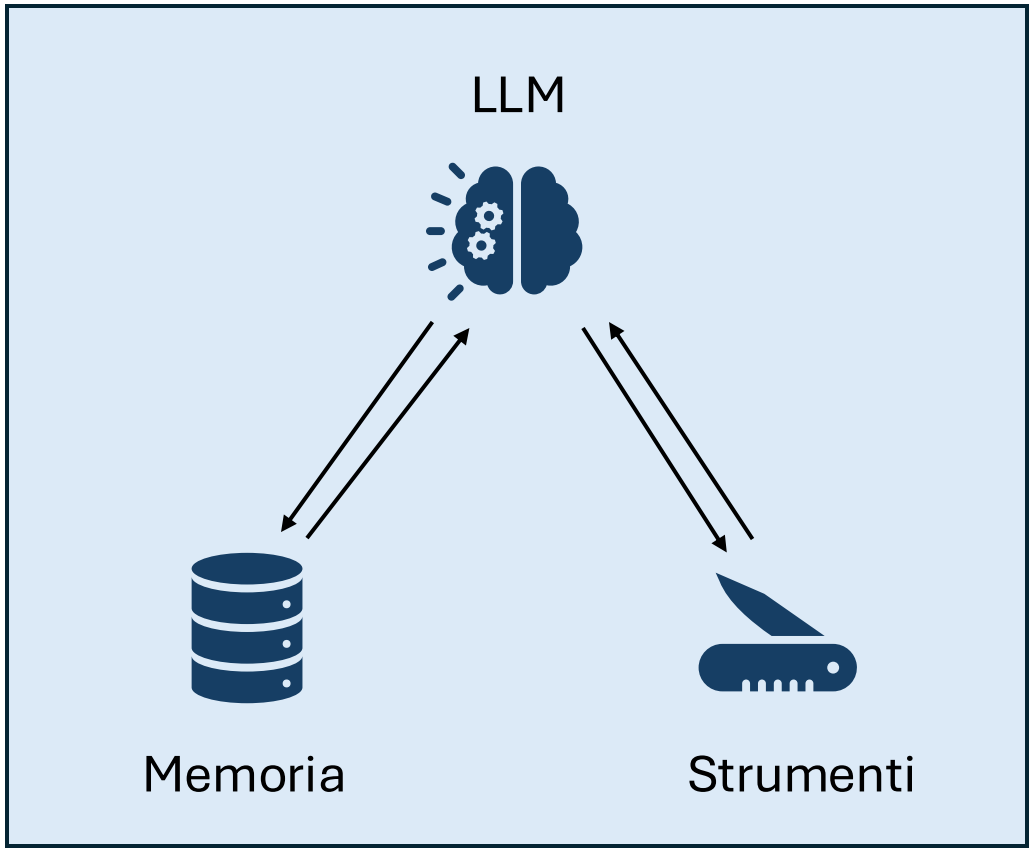
Le **intentions** rappresentano lo **stato deliberativo dell'agente**, cioè ciò che l'agente ha **deciso di fare**.



Le intenzioni sono **desires a cui l'agente si è dedicato**, almeno in parte.

Nei sistemi implementati, questo impegno si manifesta tipicamente nel fatto che l'agente **ha iniziato (o sta per iniziare) l'esecuzione di un piano**.

Sistema di Agentic AI

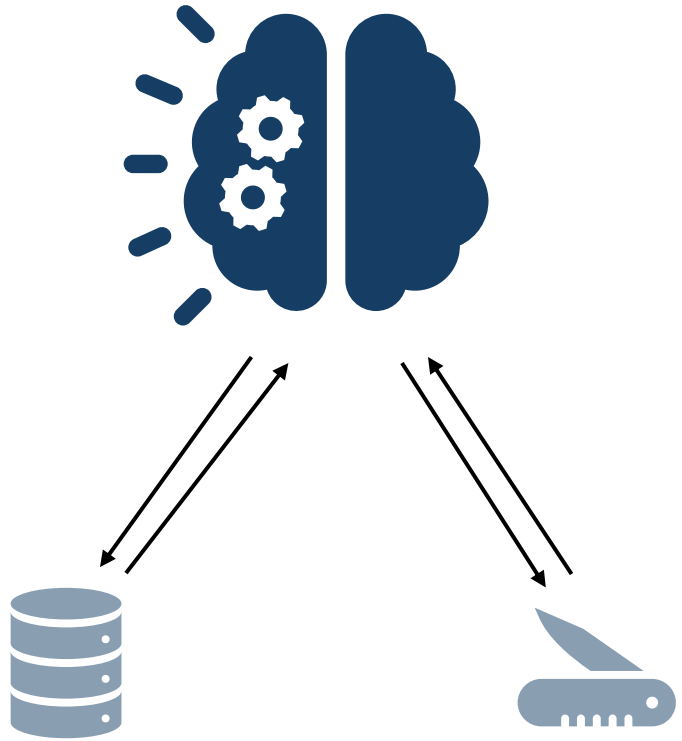


Sistema di Agentic AI: LLM

Il **Large Language Model** (LLM) riceve in input il **goal** imposto dall'utente e le **osservazioni** provenienti dall'ambiente.

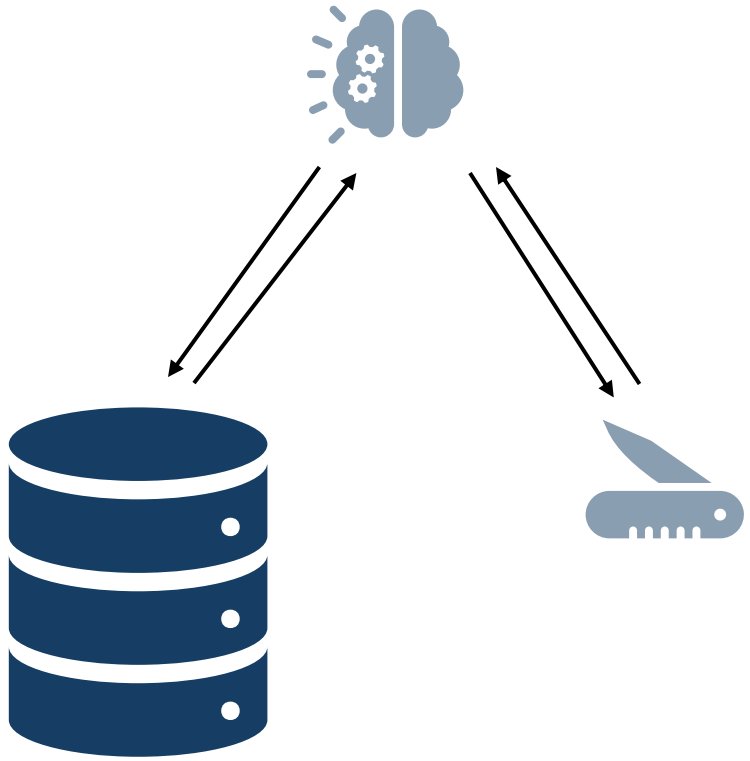
L'LLM in questo sistema deve:

- **Ragionare** con le informazioni a disposizione
- Generare il **piano**
- Decidere quali **strumenti** utilizzare



Sistema di Agentic AI: Memoria (1)

La **memoria** consente a un agente di **mantenere continuità nel comportamento** oltre la singola interazione.
In un sistema di Agentic AI, la memoria rappresenta una forma di **stato interno persistente**, utilizzata per supportare ragionamento, pianificazione e decisione.



La memoria ha il ruolo di:

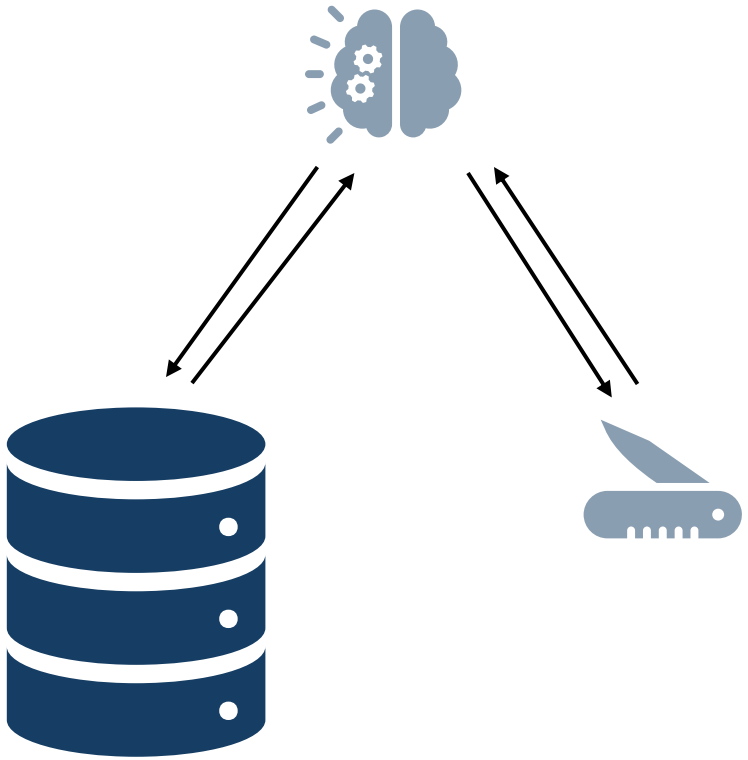
- Supportare l'aggiornamento dei **beliefs** dell'agente.
- Consente di **evitare ripetizioni** e di mantenere coerenza nel tempo.
- Facilita l'adattamento del comportamento in compiti di lunga durata.

Sistema di Agentic AI: Memoria (2)

Il sistema di Agentic AI utilizza due tipi di memoria:

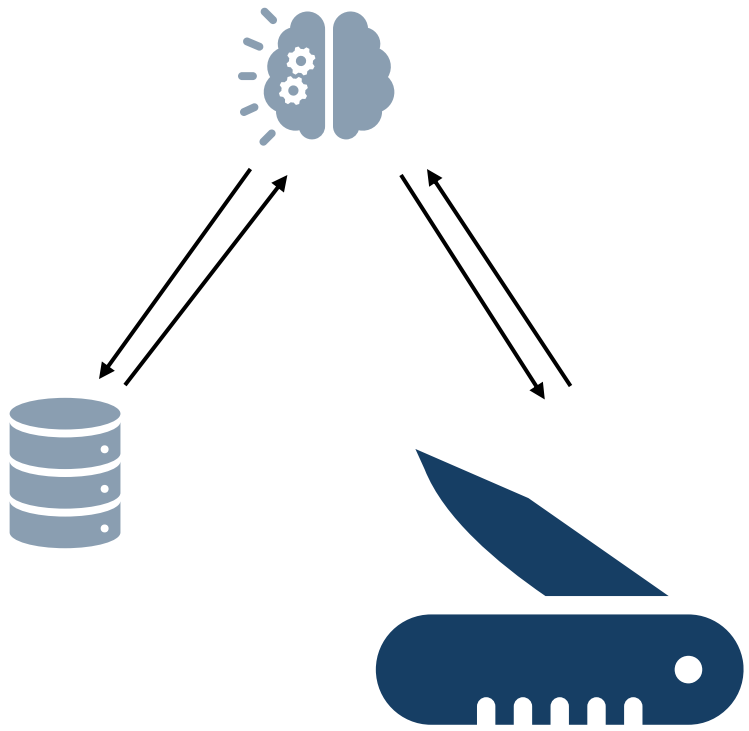
- **Memoria a breve termine**: sintetizza i concetti utili che vengono forniti in input al LLM tramite contesto (prompt)

- **Memoria a lungo termine**: immagazzina le informazioni in strutture dati come Data Base e Knoledge Base



Sistema di Agentic AI: Strumenti

La caratteristica principale degli **strumenti** utilizzati dal sistema di Agentic AI è quella di svolgere una **specifica funzione**. Ogni strumento è **indipendente** dal sistema ed è trattato come una **black-box**: riceve un input dall'LLM e ritorna il suo output.



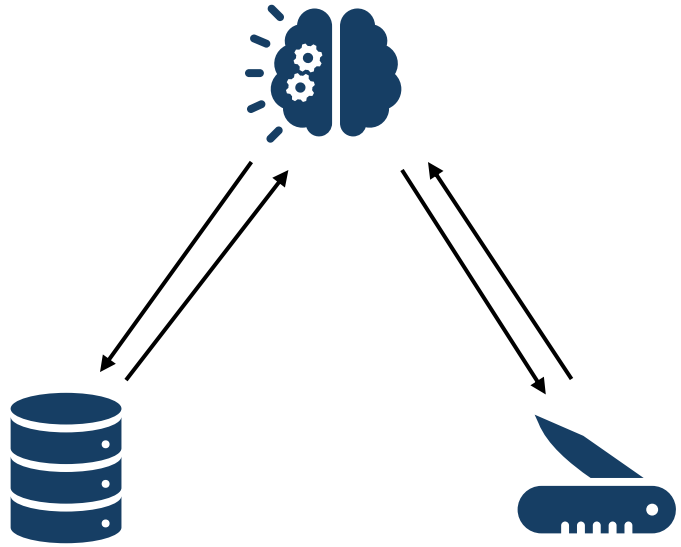
Gli strumenti utilizzati dal sistema di Agentic AI possono essere:

- API
- Pianificatori
- Database
- Altri agenti

Sistema di Agentic AI: Funzionamento

Il funzionamento del sistema si basa su un ciclo composto da tre azioni:

1. **Osserva**: Il sistema riceve delle osservazioni dall'ambiente
2. **Pensa**: Il sistema elabora queste informazioni integrandole con la propria memoria per generare un piano
3. **Agisci**: Il sistema sceglie lo strumento adatto per eseguire l'azione scelta. Quest'azione viene eseguita dallo strumento che modifica l'ambiente generando nuove osservazioni.

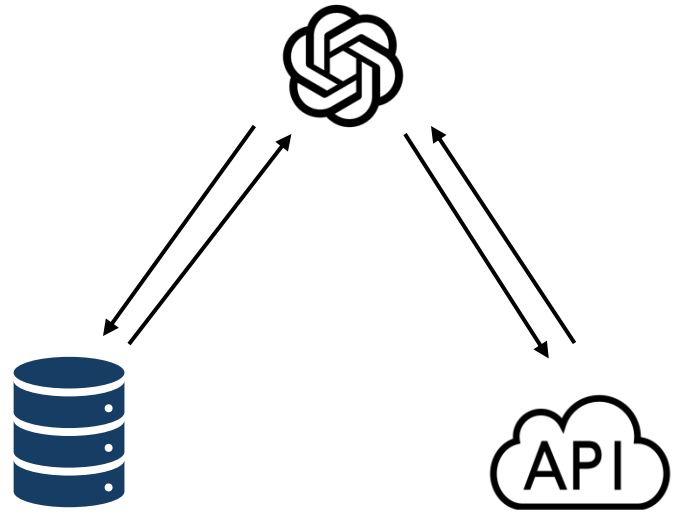


Sistema di Agentic AI: Esempio

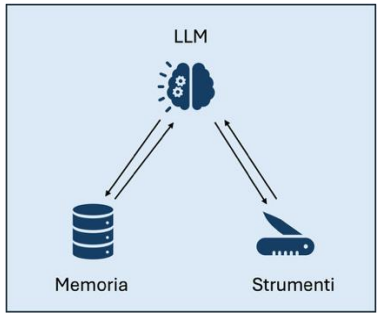
Utente: «Crea uno schema riassuntivo dell'argomento Logica spiegato nel corso di Intelligenza Artificiale tenuto in UNIBS»

Agente:

1. Scompone il task: ricerca → filtra → raggruppa → riassume
2. Utilizza strumenti web/API per ottenere le slide
3. Riassume il contenuto delle slide e produce lo schema



LLM vs Sistema di Agentic AI



Risponde a un prompt	⟷	Persegue un obiettivo
Output testuale	⟷	Azioni sull’ambiente
Nessun ciclo autonomo	⟷	Ciclo percezione–azione
Memoria limitata	⟷	Stato interno persistente
Nessun impegno	⟷	Intenzioni e piani

Sistema di Agentic AI con Agenti Multipli

I **sistemi multi-agente (MAS)** studiano insiemi di agenti autonomi che **interagiscono** all'interno di un ambiente comune. In questi sistemi, gli agenti possono **cooperare, coordinarsi o competere** per raggiungere obiettivi individuali o condivisi.

Molti sistemi di **Agentic AI** possono essere interpretati come **sistemi multi-agente**, in cui più agenti specializzati collaborano e vengono orchestrati per risolvere compiti complessi.

Concetti classici dei MAS — come **coordinamento, comunicazione e distribuzione dei compiti** — risultano quindi centrali anche nelle sistemi di Agentic AI.

Agentic AI: Riassunto

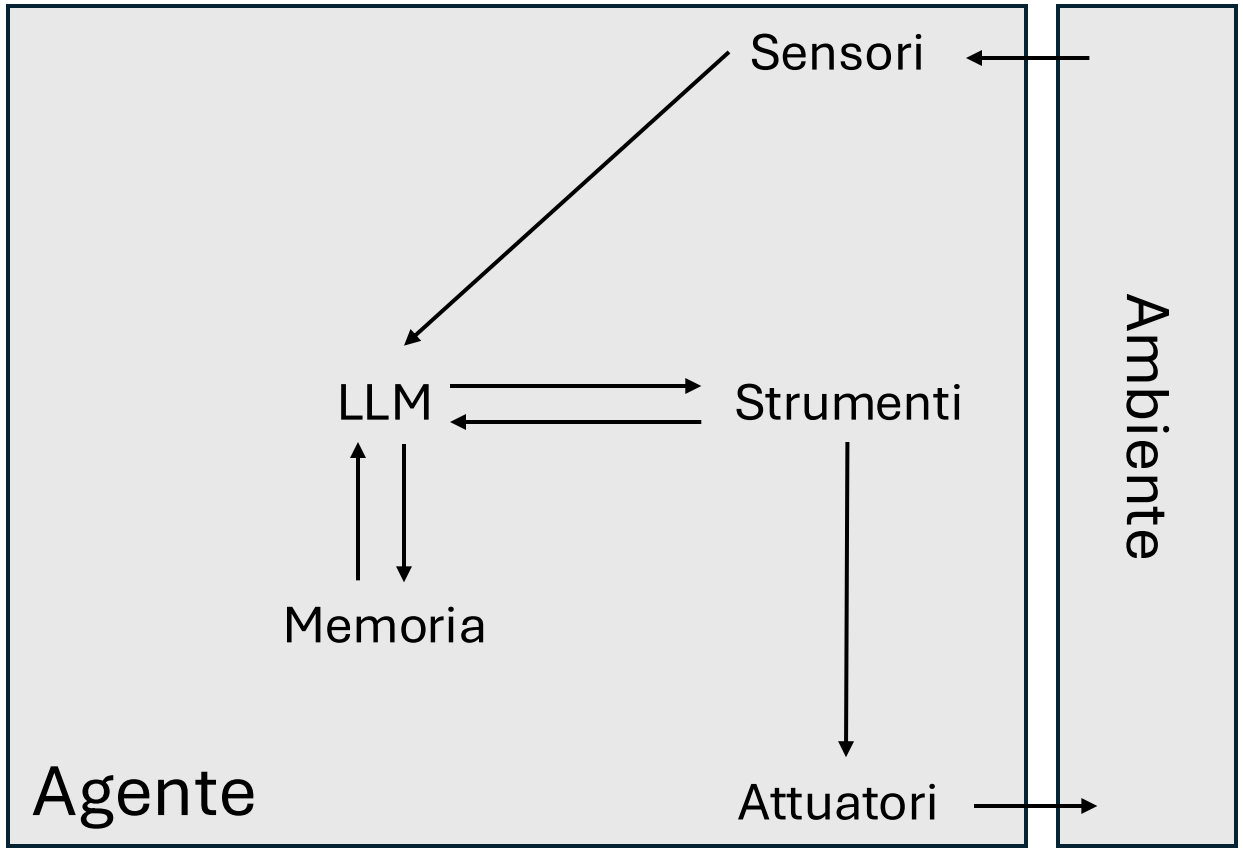
I sistemi di Agentic AI estendono i classici agenti attraverso:

- Utilizzo di LLM
- Strumenti e Memoria esterni
- Gestione di multipli agenti

Domanda

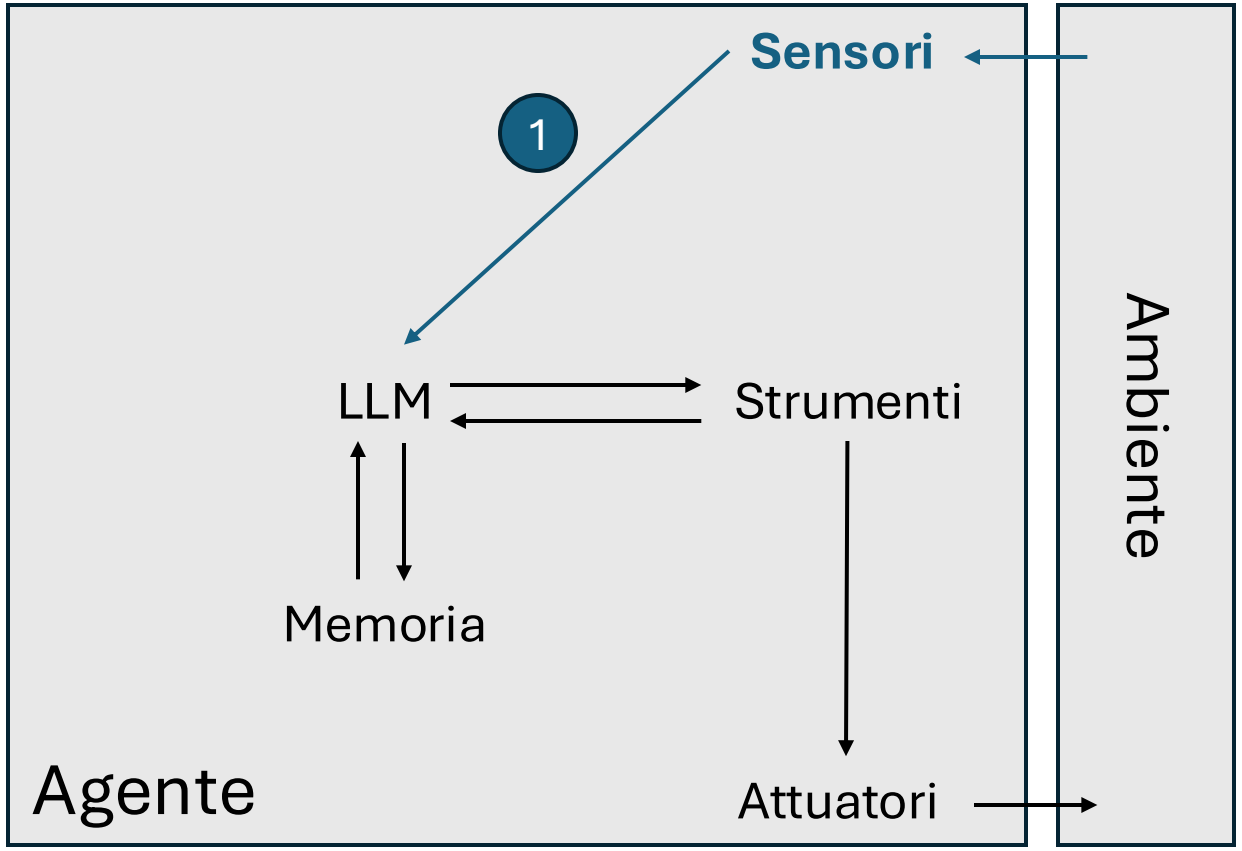
Quanto sono nuovi i concetti introdotti nell'agentic AI e quanto invece è una rivisitazione di idee già esistenti negli agenti classici?

Agente basato su LLM



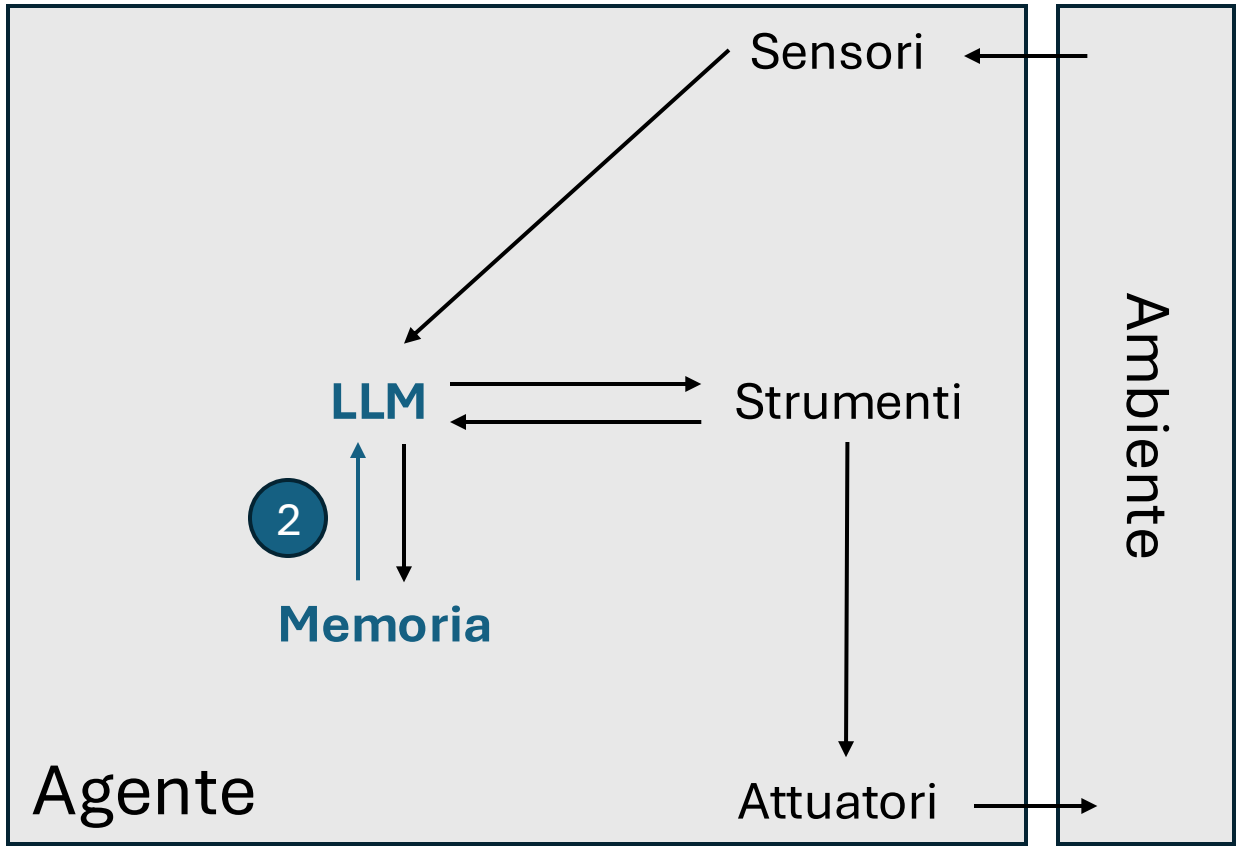
Agente basato su LLM: Ciclo di funzionamento (1)

1. **Ricevo il prossimo goal / osservazione**
2. Recupero dalla memoria le informazioni rilevanti e creo il contesto (prompt)
3. Tramite l'LLM propongo la prossima azione / sottopiano
4. Eseguo l'azione attraverso gli strumenti che ho a disposizione
5. Osservo il risultato e aggiorno la memoria



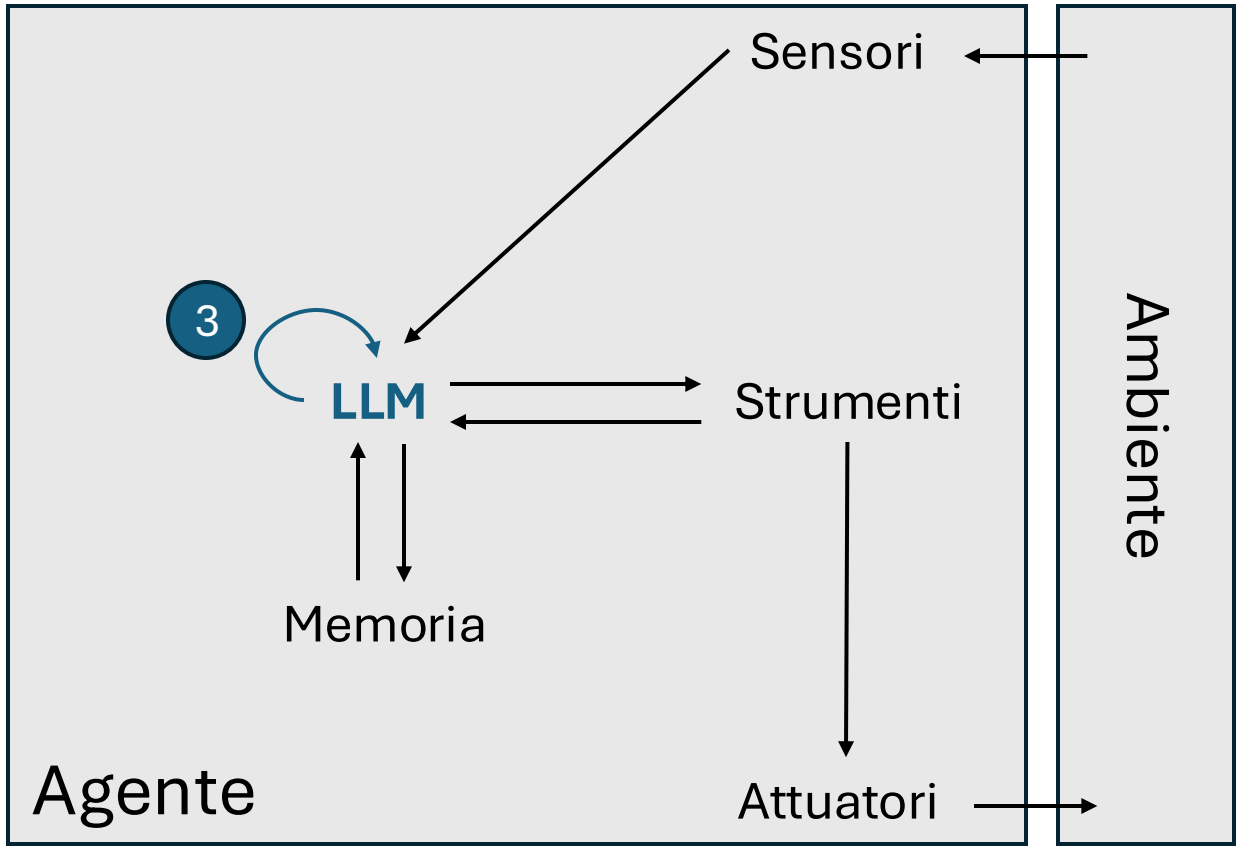
Agente basato su LLM: Ciclo di funzionamento (2)

1. Ricevo il prossimo goal / osservazione
2. **Recupero dalla memoria le informazioni rilevanti e creo il contesto (prompt)**
3. Tramite l'LLM propongo la prossima azione / sottopiano
4. Eseguo l'azione attraverso gli strumenti che ho a disposizione
5. Osservo il risultato e aggiorno la memoria



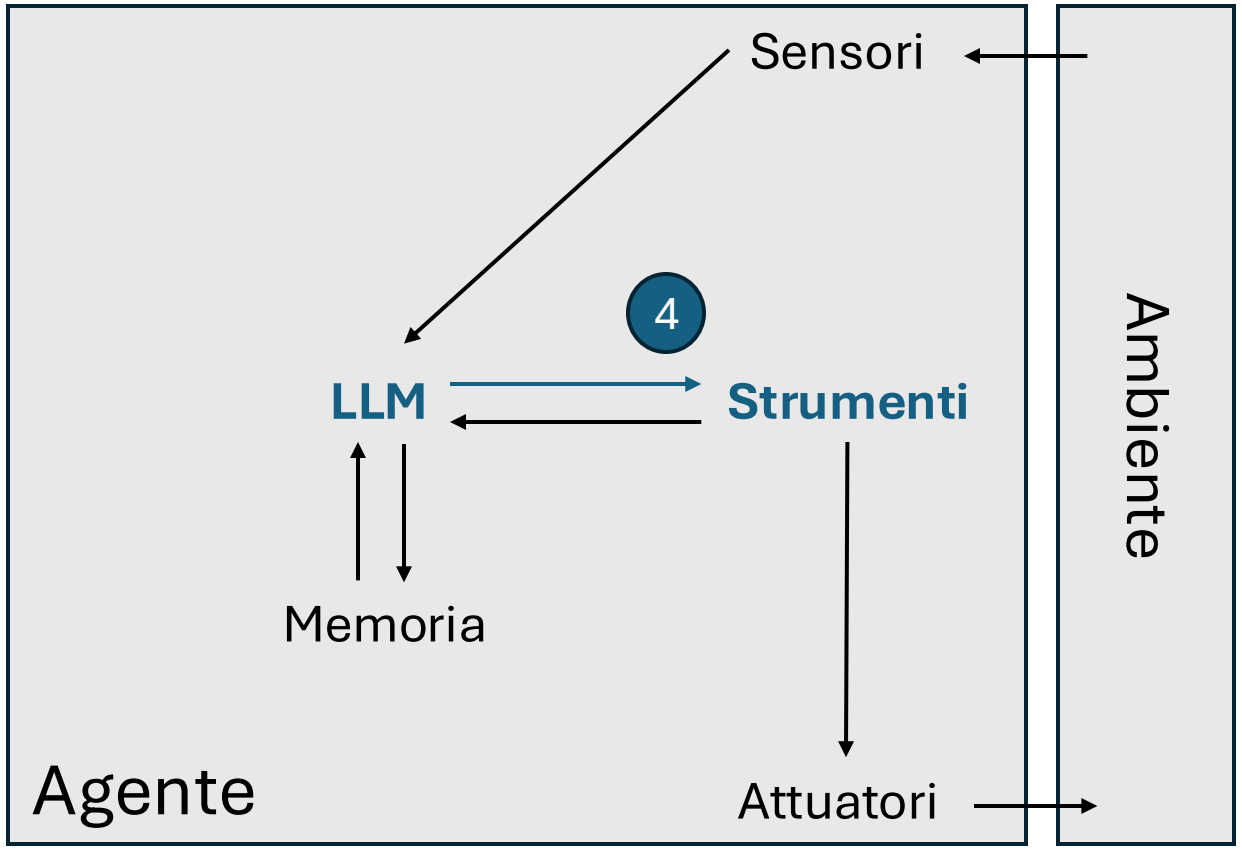
Agente basato su LLM: Ciclo di funzionamento (3)

1. Ricevo il prossimo goal / osservazione
2. Recupero dalla memoria le informazioni rilevanti e creo il contesto (prompt)
3. **Tramite l'LLM propongo la prossima azione / sottopiano**
4. Eseguo l'azione attraverso gli strumenti che ho a disposizione
5. Osservo il risultato e aggiorno la memoria



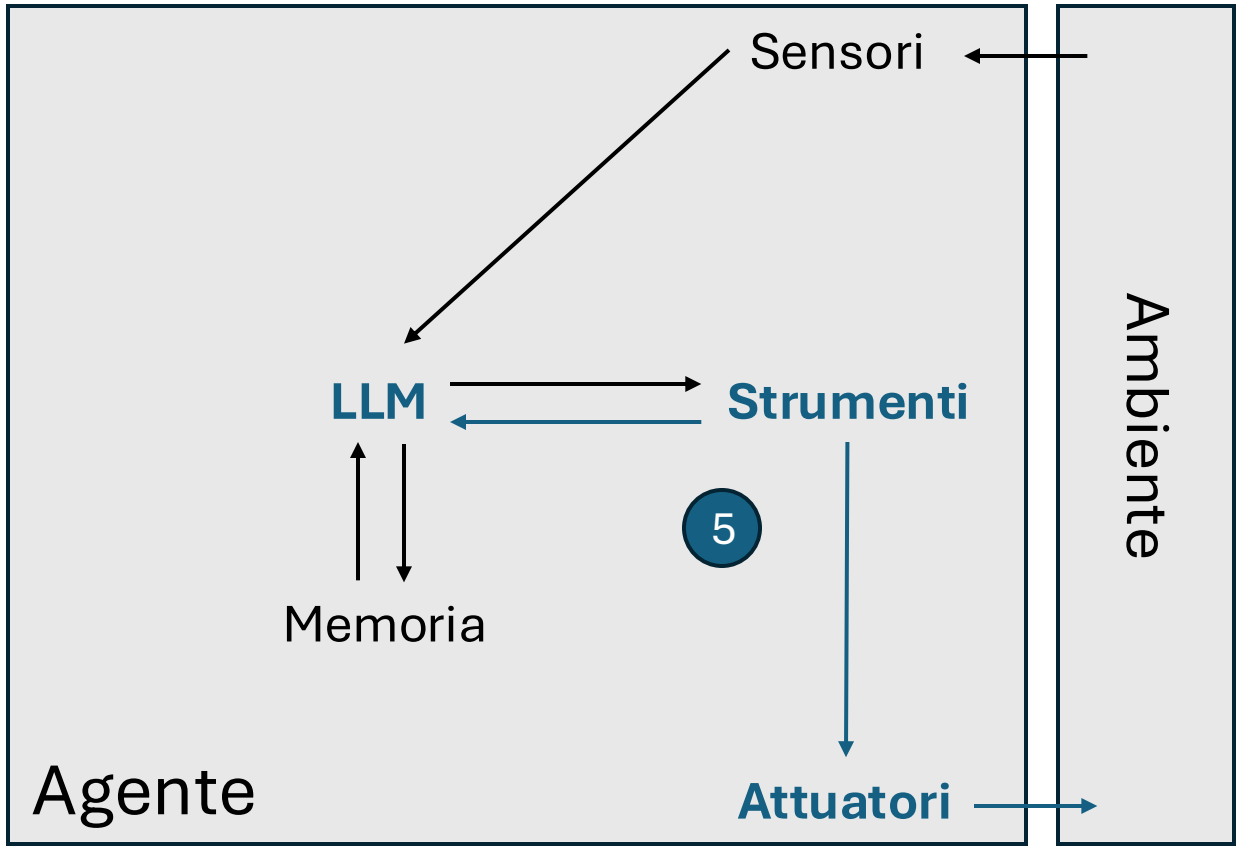
Agente basato su LLM: Ciclo di funzionamento (4)

1. Ricevo il prossimo goal / osservazione
2. Recupero dalla memoria le informazioni rilevanti e creo il contesto (prompt)
3. Tramite l'LLM propongo la prossima azione / sottopiano
4. **Eseguo l'azione attraverso gli strumenti che ho a disposizione**
5. Osservo il risultato e aggiorno la memoria



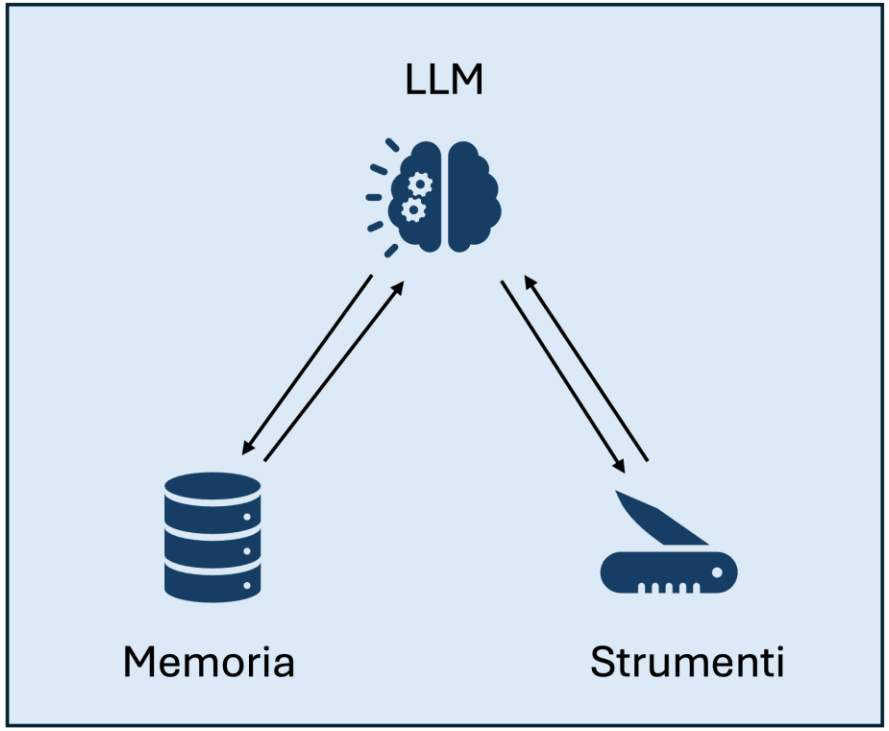
Agente basato su LLM: Ciclo di funzionamento (5)

1. Ricevo il prossimo goal / osservazione
2. Recupero dalla memoria le informazioni rilevanti e creo il contesto (prompt)
3. Tramite l'LLM propongo la prossima azione / sottopiano
4. Eseguo l'azione attraverso gli strumenti che ho a disposizione
5. **Osservo il risultato e aggiorno la memoria**



Architettura Singolo Agente con Planner

In questa architettura già vista in precedenza, abbiamo **un solo agente** che si occupa di **gestire gli strumenti** a sua disposizione e **tenere aggiornata la propria memoria** per raggiungere il proprio **goal**



Ragionare con gli LLM: ReAct [Yao et al., 2022]

Domanda

Quale strategia di prompt devo utilizzare per ottenere un'azione o un sottopiano da un LLM?

Posso utilizzare **ReAct** (*Reasoning and Acting*), un **pattern di prompting** per modelli di linguaggio che consente di **intercalare esplicitamente ragionamento e azione** durante l'esecuzione di un compito.

ReAct: Idea

L’LLM alterna esplicitamente 3 fasi:

1. **Ragionamento (Thought):** passo di ragionamento esplicito
2. **Azioni (Action):** invocazione di uno strumento o esecuzione di un’azione
3. **Osservazioni (Observations):** risultato dell’azione

ReAct: Esempio

Utente: «Crea uno schema riassuntivo dell'argomento Logica spiegato nel corso di Intelligenza Artificiale tenuto in UNIBS»

Thought: devo cercare informazioni
Action: search("slides Intelligenza Artificiale UNIBS")
Observation: risultati trovati
Thought: ora posso filtrare quelle sulla logica
…

ReAct: Pro e Contro

React è utile perché:

- Rende il **ragionamento esplicito e interpretabile**.
- Facilita l'**integrazione con strumenti esterni** (ricerca, calcolo, API).
- Riduce errori dovuti a ragionamento puramente "a priori", consentendo al modello di **verificare ipotesi tramite l'azione**.
- Implementa naturalmente un comportamento **online**, interleavando decisione ed esecuzione.

Tuttavia ha dei limiti:

- Il ragionamento è **locale** (step-by-step), non garantisce pianificazione globale ottimale.
- Non introduce **impegno a lungo termine** (assenza di intentions stabili).
- Dipende fortemente dalla qualità del prompting e degli strumenti disponibili

Ragionare con gli LLM: Pianificazione Automatica

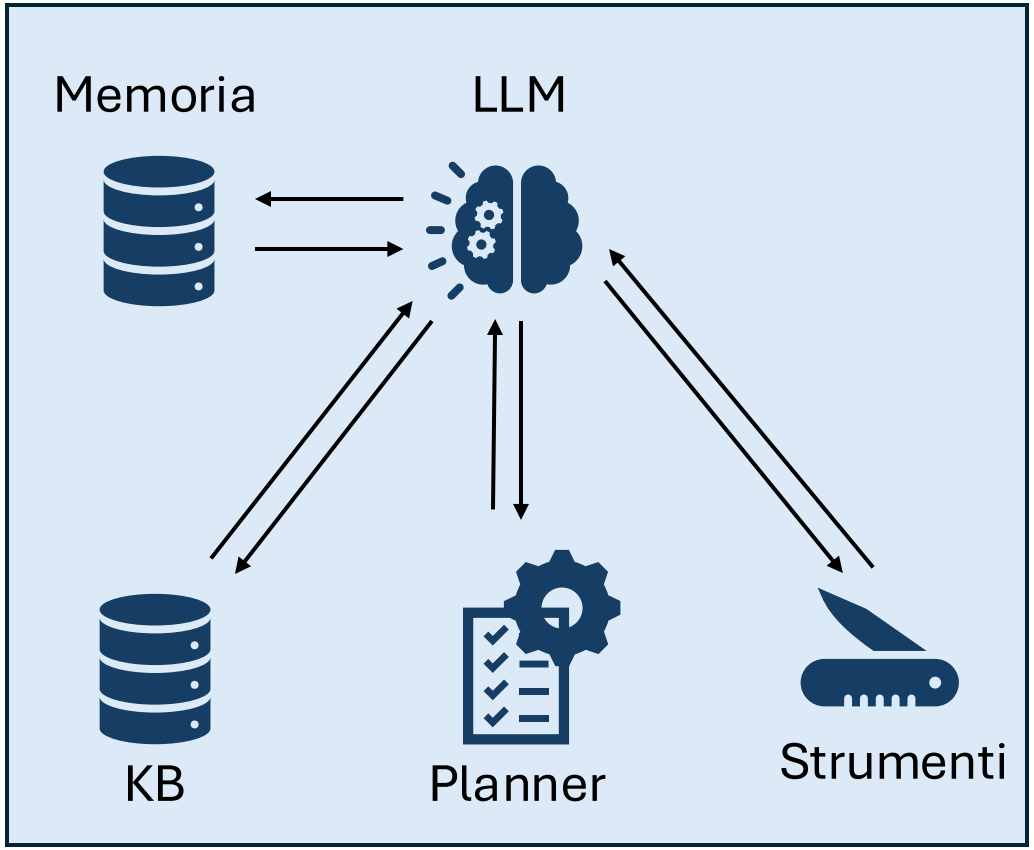
Domanda

Esiste un approccio per ottenere un'azione o un piano tramite un ragionamento a lungo termine e con garanzie di qualità?

Possiamo utilizzare la **pianificazione automatica.**

Architettura Singolo Agente con Strumenti

In questa architettura viene introdotto il **pianificatore**, che si occupa di calcolare il piano e una **knowledge base** (KB) che contiene le informazioni sul mondo (i.e. fatti, relazioni, vincoli e regole).



Pianificazione Automatica: Idea

Utilizzando un pianificatore esterno posso:

1. Trasformare le osservazioni e il mio goal in un **problema PDDL** attraverso l'LLM

2. Utilizzare il pianificatore per **risolvere** il problema

3. **Eseguire** una per volta la sequenza di azioni ottenuta

Pianificazione Automatica: Esempio

Utente: «Ho tre blocchi chiamati A, B, C sul tavolo. Ho un braccio robotico che può prendere un blocco per volta. Il robot non può prendere un blocco che sta sotto un altro. Voglio creare la torre C, B, A dove C è il blocco più in alto.»

```
(:domain blocksworls)
(:objects A B C)
(:init (on-table A) (on-table B) (on-table C) (arm-empty) (clear A) (clear B) (clear C))
(:goal (AND (on B A) (on C B)))
```

```
((pick-up B) (stack B A) (pick—up C) (stack C B))
```

Pianificazione Automatica: Vincoli

Dato che il goal è descritto tramite un **problema PDDL**, possiamo utilizzare dei **vincoli** (constraint) per:

- Rendere **espliciti vincoli architetturali** tramite invarianti (ad esempio limitando le risorse disponibili). Gli **invarianti** sono fluenti del problema che non possono variare durante l'esecuzione del piano.

- Eliminare piani che sono **non sicuri** o **impossibili**.

Pianificazione Automatica: Pro e Contro

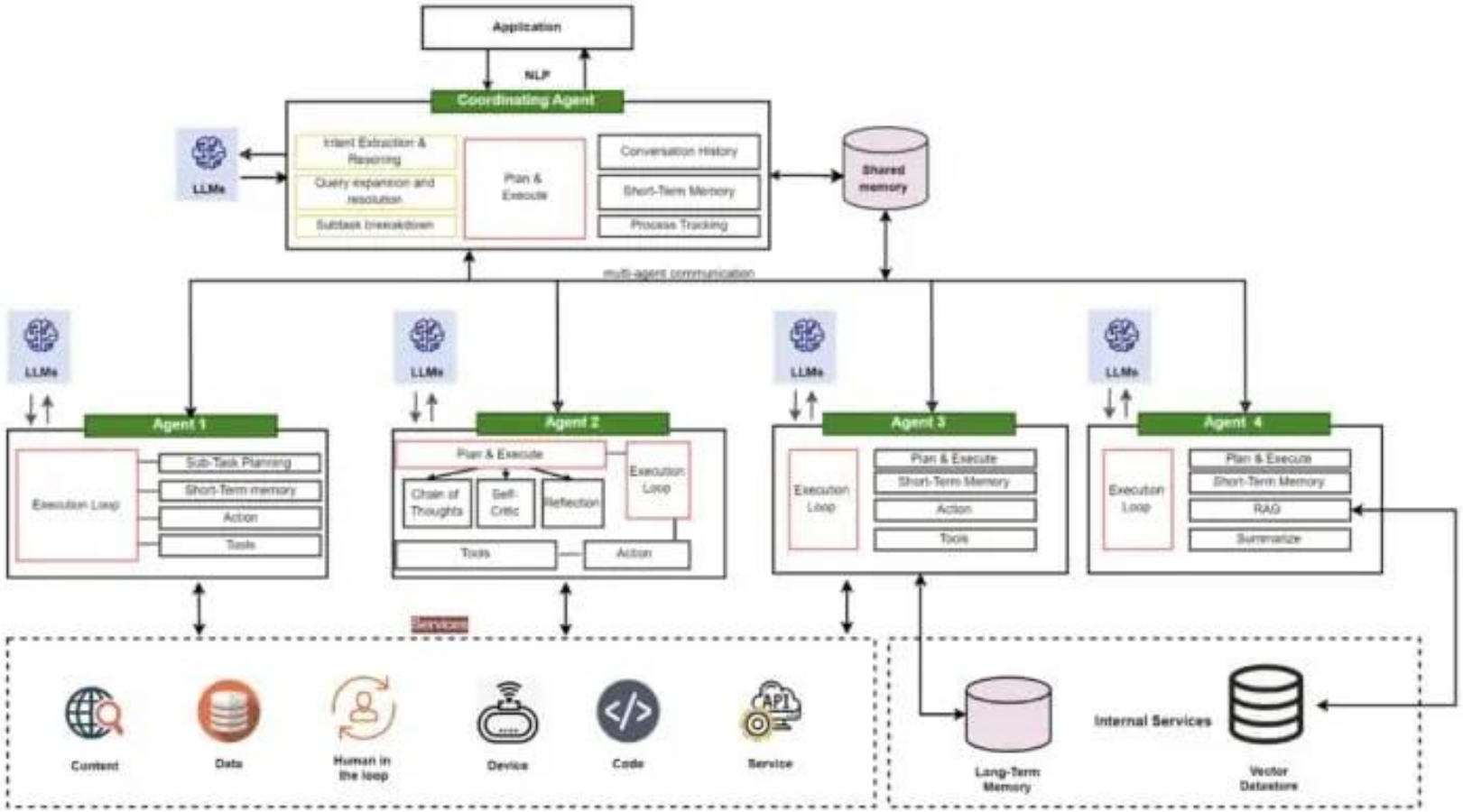
La **pianificazione automatica** permette di:

- generare sequenze di azioni **coerenti** a partire da un **modello esplicito** del dominio.
- **soddisfare formalmente** precondizioni, effetti e vincoli.
- **Riusare** modelli di dominio (es. STRIPS/PDDL)
- Pianificare a **lungo termine**

Tuttavia anche la pianificazione ha dei limiti:

- può essere **onerosa** in domini ampi o dinamici.
- è necessario definire stati, azioni e vincoli **in modo formale**.
- i piani possono diventare rapidamente **obsoleti** in ambienti altamente dinamici o incerti.
- richiede un **coordinamento accurato** con LLM.

Architettura Multi-Agente



Fonte: OWASP.org

Architettura Multi-Agente: Ruoli

Nell'architettura multi-agente, i diversi agenti possono assumere **ruoli diversi** (pianificatore, esecutore, critico, ...)

Solitamente gli agenti **interagiscono** attraverso uno **scambio di messaggi**.

Questa architettura permette una **maggiore specializzazione** degli agenti e può garantire **maggiore robustezza**. Tuttavia, questi vantaggi richiedono la **coordinazione** dell'intero sistema e la **gestione dei messaggi** scambiati tra gli agenti.

Architettura Multi-Agente: Pattern di Gestione

Ci sono 3 principali **pattern** per gestire un'architettura multi-agente:

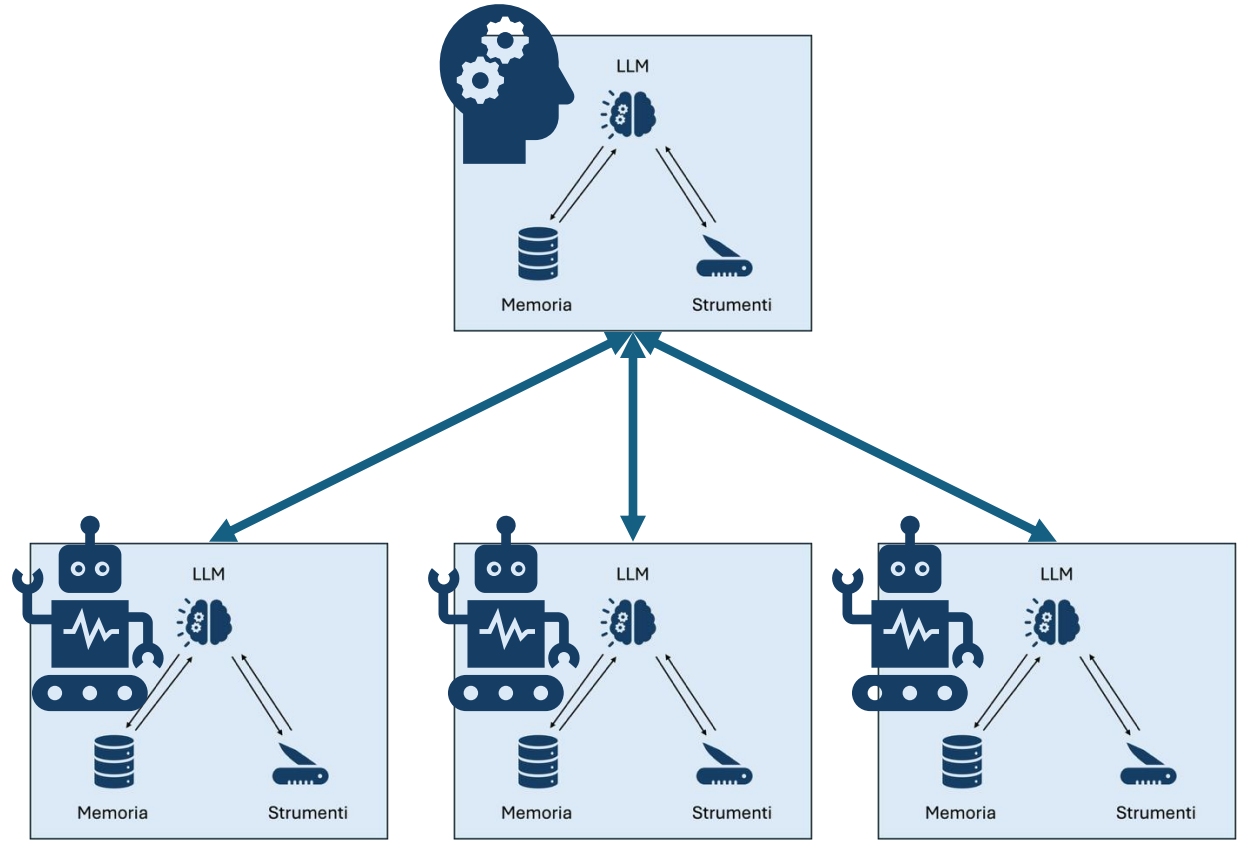
- Orchestrator + workers
- Peer-to-peer
- Hierarchical

Orchestrator + Workers Pattern

In questo pattern un agente principale (chiamato **orchestrator**) gestisce e comunica con una serie di agenti specializzati (chiamati **worker**).

Ogni worker svolge un task specifico e comunica solamente con l'orchestrator

Pattern che permette una gestione molto semplice dal punto di vista della comunicazione ma che dipende molto dall'orchestrator

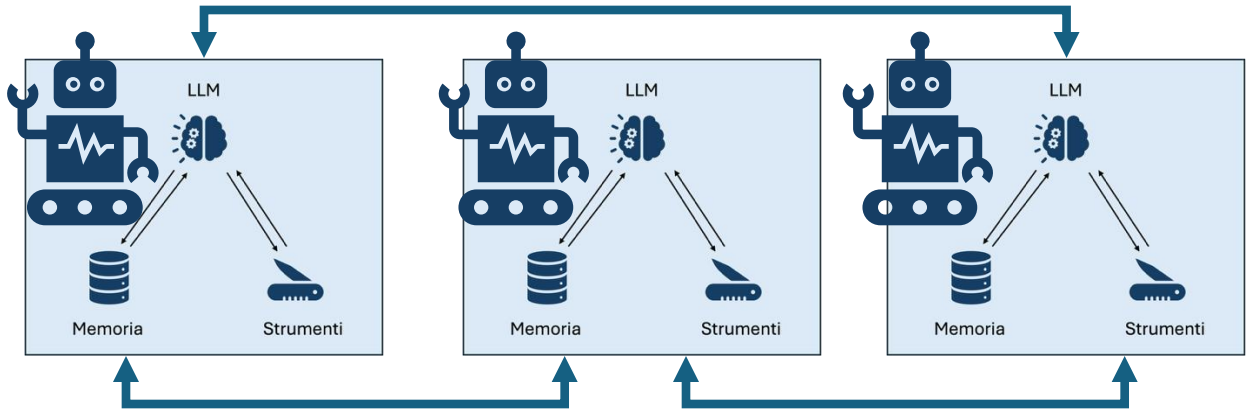


Peer-to-Peer Pattern

In questo pattern tutti gli agenti **comunicano** e si **coordinano** tra di loro.

Ogni agente svolge uno o più **sotto-task** e **interagisce con gli altri** per raggiungere il goal finale. Ogni sotto-task può essere svolta in parallelo da più di un agente (**voting system**)

La coordinazione tra i singoli agenti è **complessa** ma rende il sistema complessivamente **robusto**.

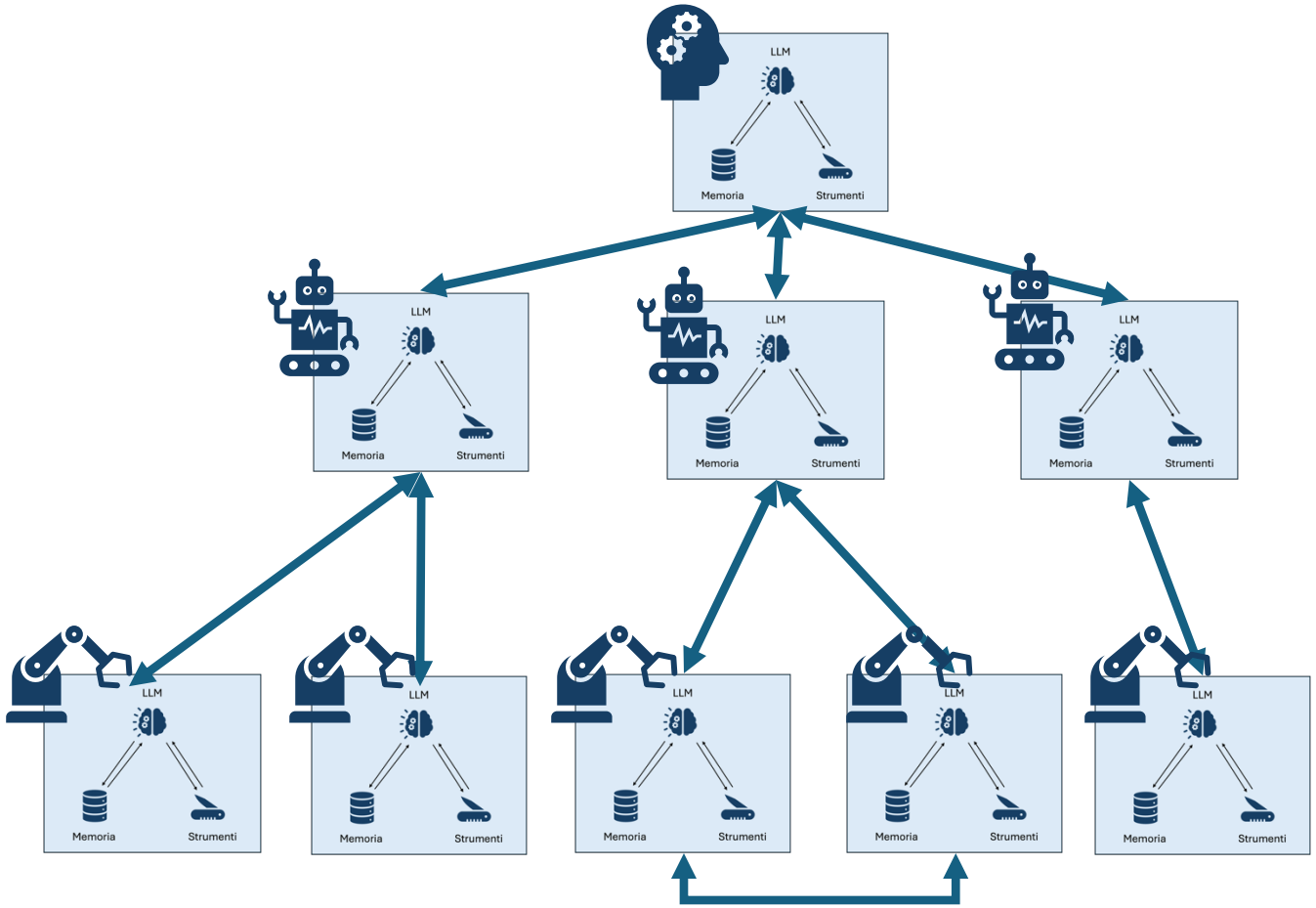


Hierarchical Pattern

In questo pattern il coordinatore (**orchestrator**) assegna i compiti a degli agenti supervisori (**supervisor**) che, a loro volta, si occupano di assegnare i compiti agli agenti specializzati (**workers**)

Possono esserci **diversi livelli** di supervisione.

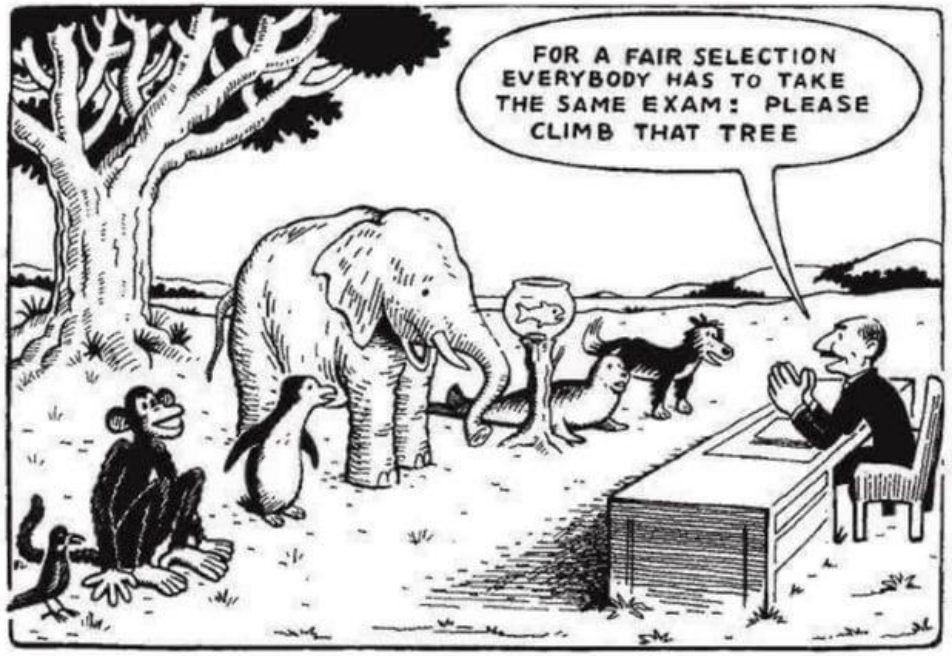
Architettura pensata per **task complessi** in cui bisogna avere sia un buon coordinamento che una buona comunicazione



Valutare un Sistema di Agentic AI

I sistemi di **Agentic AI** operano in modo autonomo, su compiti complessi e di lunga durata.

Comprendere **come valutarne il comportamento** è essenziale per garantire **affidabilità, sicurezza e controllo**, ma rappresenta una **sfida aperta**.



Fonte: reddit.com

Valutare un Sistema di Agentic AI: Metriche

Alcune metriche per valutare un sistema di Agentic AI sono:

- **Task success rate:** percentuale di compiti completati correttamente
- **Tempo e costo:** risorse computazionali e operative utilizzate
- **Robustezza:** resistenza a rumore, input ambigui e adversarial prompt.
- **Sicurezza:** violazione di vincoli, comportamenti non consentiti, azioni pericolose

Valutare un Sistema di Agentic AI: Benchmark

In aggiunta, è possibile valutare i sistemi di Agentic AI su **test suite** costruite per valutare diverse capacità specifiche tra cui:

- Ingegneria del software
- Navigazione web
- Uso di strumenti ed API
- Integrazione con vari sistemi operativi
- Coordinamento multi-agente

Valutare un Sistema di Agentic AI: Problemi Aperti

Ci sono alcune proprietà dei sistemi di Agentic AI che sono difficili da catturare e che quindi sono tutt'ora oggetto di ricerca e di un metodo più rigoroso di valutazione.

Alcune di queste proprietà sono:

- **Comportamento a lungo termine:** valutare agenti che operano con molte iterazioni
- **Proprietà emergenti:** valutare interazioni non previste tra componenti ed agenti
- **Impatto sociale ed etico:** valutare effetti indiretti, responsabilità e fiducia.

Incertezza nei Sistemi di Agentic AI (1)

I sistemi di Agentic AI devono fare i conti con l'**incertezza**, che in questi sistemi nasce da diverse fonti:

- **Osservabilità parziale:** i dati ricevuti possono essere parziali o incompleti, possono essere sottointesi o non contenere elementi che l'utente da per scontati (common-sense knowledge)
- **Azioni stocastiche:** malfunzionamento degli strumenti, errori di rete, impredicibilità dell'ambiente
- **Incertezza del modello:** risposte impredicibili e sbagliate dell'LLM (allucinazioni), strumenti imprecisi o non corretti

Incertezza nei Sistemi di Agentic AI (2)

La presenza dell'incertezza in questi sistemi va inizialmente ad influire sulle **belief** (credenze) del sistema. L'agente **non è incerto su ciò che vuole fare**, ma su **ciò che crede vero.**

Infatti tutte le fonti di incertezza viste vengono inizialmente registrate dal modello e **intaccano la sua rappresentazione** del mondo.

Tuttavia questa incertezza non si ferma solo alle belief, ma si propaga poi ai **goal**, alle **intentions** e ai **piani**, influenzando le decisioni e il comportamento complessivo dell'agente.

Incertezza nei Sistemi di Agentic AI (3)

È possibile attuare alcune strategie architetturali per mitigare l'effetto dell'incertezza:

- **Aggiornamento continuo delle beliefs:** integrazione dei feedback dell'ambiente e revisione dello delle belief
- **Pianificazione robusta e ripianificazione:** generazione di piani flessibili e capacità di adattarsi ai cambiamenti inattesi
- **Integrazione di conoscenza affidabile:** uso di Knowledge Base verificate per ridurre ambiguità e allucinazioni
- **Human-in-the-loop:** prevedere alcuni step di controllo tramite intervento umano, in particolare in caso di alta incertezza o alto rischio

Sicurezza nei Sistemi di Agentic AI (1)

I sistemi di Agentic AI operano in modo autonomo e interagiscono con ambiente e strumenti esterni, introducendo rischi per la sicurezza:

- **Data Exfiltration:** perdita non intenzionale di informazioni sensibili tramite LLM e strumenti. Data Exfiltration può capitare, ad esempio, quando un LLM ha accesso a dei dati privati e li inserisce in una risposta pubblica.
- **Prompt Injection:** manipolazione del comportamento dell'agente tramite input malevoli. Questi input condizionano il comportamento dell'agente e possono fargli compiere azioni pericolose o malevole.
- **Runaway loop:** cicli di azioni non controllati che consumano risorse senza raggiungere l'obiettivo
- **Azioni non autorizzate:** uso improprio di strumenti e API

Sicurezza nei Sistemi di Agentic AI (2)

È possibile migliorare la **sicurezza** dei sistemi di Agentic AI, come avviene per l'incertezza, adottando alcune strategie architetturali:

- **Separazione dei ruoli:** distinguere chiaramente ragionamento, pianificazione ed esecuzione.
- **Audit logs:** registrare tutta la storia del sistema includendo i log e i dettagli su quando ogni agente ha svolto un compito.
- **Human-in-the-loop:** come per la gestione dell'incertezza, l'intervento umano in casi ad alto rischio possono migliorare di molto la sicurezza del sistema.

Sicurezza nei Sistemi di Agentic AI (3)

Da un punto di vista **pratico**, invece, alcuni accorgimenti per aumentare la sicurezza possono essere:

- **Vincoli e controlli:** limitare alcune azioni, i costi e il numero di iterazioni.
- **Esecuzione "sandbox" degli strumenti:** permettere l'esecuzione degli strumenti in un ambiente isolato (sandbox).
- **Agenti e critici di controllo:** creare alcuni agenti che avranno il compito di controllare le informazioni sia in entrata al sistema che in uscita.

Sicurezza nei Sistemi di Agentic AI (3)

Da un punto di vista **pratico**, invece, alcuni accorgimenti per aumentare la sicurezza possono essere:

- **Vincoli e controlli:** limitare alcune azioni, i costi e il numero di iterazioni.
- **Esecuzione "sandbox" degli strumenti:** permettere l'esecuzione degli strumenti in un ambiente isolato (sandbox).
- **Agenti e critici di controllo:** creare alcuni agenti che avranno il compito di controllare le informazioni sia in entrata al sistema che in uscita.

Frameworks per Sistemi di Agentic AI

Alcuni framework disponibili per sperimentare con gli agenti sono:

- CrewAI
- AG2 (prima chiamato AutoGen)
- LangChain + LangGraph
- n8n