


AGENTS ARCHITECTURES

PARADIGMA AGENT ORIENTED

- (1) **REAZIVITÀ**: Abilità di agire in base alle percezioni
- (2) **PROATTIVITÀ**: Abilità di prendere iniziative appropriate per raggiungere il suo scopo
- (3) **ABILITÀ SOCIALE**: Abilità di comunicare con altri agenti

AGENT ORIENTED VS OBJECT ORIENTED

	Oggetto	Agente
Sistemi di definizione parametri	Nessuna Restrizione	Credenze, impegno nelle scelte, capacità,
Processo di computazione	Passaggio di Messaggi e Metodi di Risposta	Passaggio di Messaggi e Metodi di Risposta
Tipi di messaggio	Nessuna Restrizione	Informa, richiede, offre, ecc.
Restrizioni sui metodi	Nessuna	Onestà, consistenza, ecc.

Gli agenti non invocano metodi su un altro ma richiedono azioni per essere eseguiti
 Gli oggetti non hanno comportamenti autonomi flessibili
 Ogni agente ha il suo processo di controllo

ARCHITETTURE

AGENTI CON STATO

Uno **STATO INIZIALE INTERNO** è aggiornato dalle percezioni, quello attuale determina le azioni

AGENTI CON RAGIONAMENTO DEDUTIVO

"Gli agenti compiono decisioni riguardo cosa fare attraverso la **manipolazione di simboli**. La sua espressione più pura propone che l'agente usi **esplicito ragionamento logico** in modo da decidere cosa fare"

Sono dimostratori di teoremi, rappresentano simbolicamente ambiente, comportamenti, obiettivi, ... e, hanno un insieme di azioni da eseguire che sono soddisfatte dalle percezioni, cioè formule logiche, da controllare tramite **DEDUZIONI LOGICHE**

→ AGENTI DECUBERATIVI

- set di credenze espresse come FORMULE LOGICHE
- set di AZIONI
- set di REGOLE DI DEDUZIONE
- FUNZ. DI SEZIONE, possono essere più azioni eseguibili

AGENTI CON RAGIONAMENTO PRATICO

"Gli agenti usano ragionamenti pratici (attraverso le azioni, non attraverso le credenze) – credenze / desideri / intenzioni."

Ragionamento diretto ad AZIONI anziché credenze

• BDI

- **BELIEFS** caratteristiche dell'ambiente o ragionamenti dati da Percezioni sensoriali (componente **INFORMATIVO** del BDI)
- **DESIRS** info sui GOAL, Problemi e Priorità associate (componente **MOTIVAZIONALE** del BDI)
- **INTENDINGS** rappresenta l'azione in corso, cioè l'ultimo output della **SELECTION FUNCTION** (componente **DECUBERATIVO** del BDI) (**Desiderio Fattibile**)
- **FUNZIONE DI SELEZIONE**
- **SEMANTICA**
 - **X Bel b** se b è vero in possibile mondo raggiungibile da X
 - **X Des d** se d è in qualche mondo che X vuole raggiungere
 - ↳ non è detto che il mondo sia raggiungibile (no omniscienza sociale)
 - ↳ i **GOAL** sono un subset di DESIRS **consistenti e raggiungibili**
 - **X Int p** se p è in tutti i **CAMMINI INTESI** (cammini che X sta seguendo per raggiungere p)
 - ↳ l'insieme può essere vuoto ⇒ **intenzione insoddisfacibile**

- **COMMITMENT (RESPONSABILITÀ)** Gli agenti che persistono nelle loro intenzioni (fondi solidarietà)
 - Sono detti **COMMITTED** per tali intenzioni.
 - ↳ Questa proprietà è importante perché così gli altri sanno che non smetterà di fare nient'altro all'improvviso

• RAGIONAMENTO PRACTICO = DECUBERAZIONE + RAGIONAMENTO MEANS-END

① **Deliberazione** → OUTPUT: **INTENTION**; Divisa in due fasi

- Generazione di Opzioni: genera alternative
- Filtraggio: sceglie le alternative valide

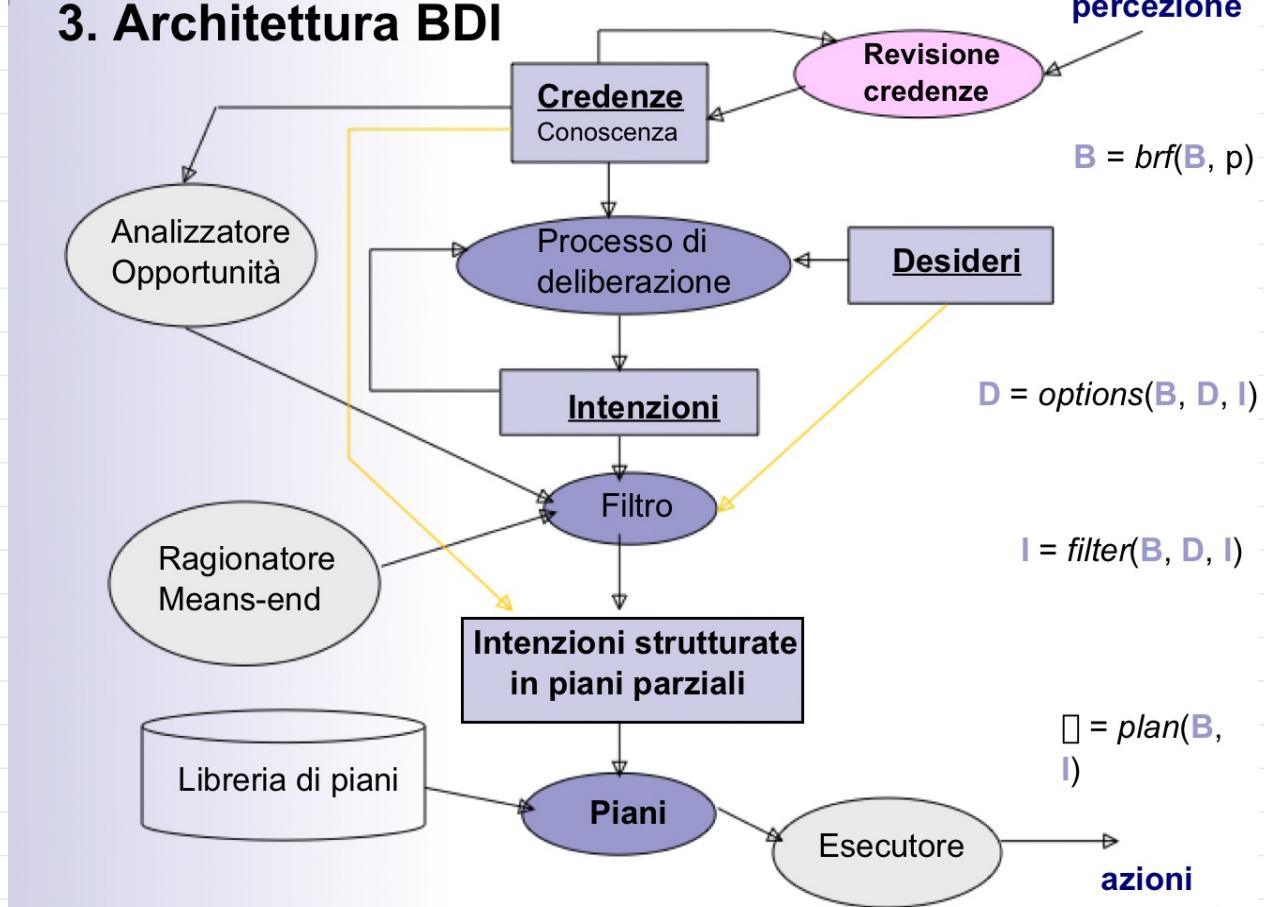
DESIRS = options (B, i)

INTENDINGS = filter (B, Di)

NOTA Un'intenzione che passa il filtro è **COMMITTED**

② **Rag. Means-End** → PIANIFICAZIONE, VALUTAZIONE e SELEZIONE PIANO

3. Architettura BDI



• Quanto un Agente può dedicarsi a una **COMMITTED INTENTION**?

- ① **BELIEF COMMITMENT** → Continua finché la raggiunge
- ② **SINGLE-MINDED** " → Continua finché è possibile ⇒ Si ferma a **DISCERNERE**
- ③ **OPEN-MINDED** " → Continua finché CREDUTO possibile ⇒ **DISCERNERE** dopo ogni azione

⇒ Come sceglie ogni quanto riconoscere le **INTENZIONI**?

↳ **CONTROLLO A METACIVELLO** dopo ogni azione mi chiedo se riconoscere le intenzioni

• PROCEDURAL REASONING AGENTS (PRAs) (IMPLEMENTAZIONE BDI)

- DB di **Credenze**
- Insieme di **Obiettivi**
- Libreria di **piani** → per raggiungere obiettivo o Reagire a eventi
- Struttura di **intenzioni** → Contiene i piani scelti
- Area di **conoscenza (KA)** → contiene procedure per eseguire i piani, Garantisca che **Condiz. di convocazione** corso con **PASSI PROCEDURALI**

EVENTI TRIGGER

• AGENT STEAK(L) (UN GUARIGLIO INGICO)

- ↳ Linguaggio basato su BDI
- **JASON** → INTERPRETE
- Pone **LIMITAZIONI** ai BDI
 - (1) Ogni regola definisce parte di un piano → **Piani fatti da regole**
 - (2) No selezione Intenzioni, No chiaro concetto di **DEDICAZIONE**

• Formato da:

- ATOMO DI CREDENZA (o la sua negazione)
- OBIETTIVO (GOAL); stato del sistema che si vuole raggiungere
 - DA raggiungere (!) → sottopiano
 - TEST (?) → ritorna una falsificazione con le credenze e false

- EVENTI ATTIVABILI (TRIGGERS)

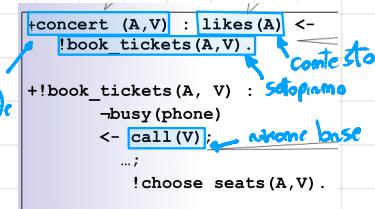
- AGGIUNTA DI CREDENZE/OBJETTIVI; (+)
- REMOZIONE " " " (-)

- PIANI: Libratoria di piani

↳ usati in risposta ad eventi di UPDATE dei BEGRES e da nuovi GOAL

- coded by developer offline, in advance
- give the agent information about
 - how to respond to events
 - how to achieve goals
- plan structure:
 - event
 - context
 - body

triggerCondition :
context <-
body.



- INTENZIONI: piani scelti per l'esecuzione

loop

- agent receives *events*, which are either
 - external (from the environment, from perceptual data)
 - internally generated
- tries to *handle* events by looking for *plans* that *match*
- the set of plans that match the event are *options/desires*
- chooses one plan from its desires to execute: becomes committed to it -- an *intention*
- as it executes a plan may generate new events that require handling

AGENTE PURAMENTE REATIVO Uno stato dell'ambiente elenca una **PERCEZIONE** grazie ai sensori,

l'insieme delle percezioni definisce le azioni

• SUBSUMPTION ARCHITECTURE

- IDEE GIGANTE

- **SITUAZIONISMO/INCORPORAZIONE** → L'intelligenza è nel mondo non in sistemi separati o indipendenti dal mondo

- **INTEL. DI EMERGENZA** → Comportamenti intelligenti dipendono da l'interazione con il mondo

- REGI GIGANTE

- **INTEL. SENZA RAPPRESENTAZIONE** → Non serve rappresentazione esplicita per descrivere comportamenti intelligenti

- **INTEL. SENZA VACUONAMENTO** → No ragionamento astratto esplicito

• **CORRASIA DI COMPORTAMENTI** Percezioni: mappati su le rispettive azioni

• **STRATEGIASIONE REALE**: gli strati più bassi (AERONI BASE come ostacoli ad es.) hanno la precedenza (cioè subsumono gli altri) → **REGOLAZIONE DI MIGRAZIONE**

• **STIGMERGIA**: Comunicazioni tramite l'ambiente

- vantaggi
 - semplice
 - economico
 - trattabile computazionalmente
 - robusto contro fallimenti
- svantaggi
 - gli agenti agiscono nel breve termine in quanto usano solo informazioni locali
 - nessun apprendimento
 - come ingegnerizzare questi agenti? Difficile se interagiscono più di 10 regole
 - nessun metodo formale di analisi e predizione

BUNS

```

B := B0;
I := I0;
while true do
    get next percept ρ;
    B := brf(B, ρ);
    D := options(B, I);
    I := filter(B, D, I);
    π := plan(B, I);
    while not empty(π) do
        α := hd(π);
        execute(α);
        π := tail(π);
        if sound(π, I, B) then
            π := plan(B, I)
        end-if
    end-while
end-while

```

Stesso di
Piano

PIANO: sequenza di azion

α = {a, b, ...}

Si un percorso è incontrato se i
Primo è ancora possibile

Reattività,
riplanificazione

SINGLE Minded

```

B := B0;
I := I0;
while true do
    get next percept ρ;
    B := brf(B, ρ);
    D := options(B, I);
    I := filter(B, D, I);
    π := plan(B, I);
    while not empty(π)
        or succeeded(I, B)
        or impossible(I, B) do
            α := hd(π);
            execute(α);
            π := tail(π);
            get next percept ρ;
            B := brf(B, ρ);
            if not sound(π, I, B) then
                π := plan(B, I)
            end-if
        end-if
    end-while
end-while

```

Lascia intenzioni
che sono impossibili
o sono state raggiunte

Si forma anche se raggiunge l'obiettivo
(primo obiettivo) il primo o si stava
impossibile

Reattività,
riplanificazione

OPEN Minded

```

B := B0;
I := I0;
while true do
    get next percept ρ;
    B := brf(B, ρ);
    D := options(B, I);
    I := filter(B, D, I);
    π := plan(B, I);
    while not (empty(π)
        or impossible(I, B))
        or succeeded(I, B) do
            α := hd(π);
            execute(α);
            π := tail(π);
            get next percept ρ;
            B := brf(B, ρ);
            D := options(B, I);
            I := filter(B, D, I);
            if not sound(π, I, B) then
                π := plan(B, I)
            end-if
        end-if
    end-while
end-while

```

Dedizione a mente aperta

A differenza di Piano raccomanda
le intenzioni

get next percept ρ;
B := brf(B, ρ);
if not sound(π, I, B) then
 π := plan(B, I)

COMMUNICATION AMONG AGENTS

ACLS (AGENT COMMUNICATION LANGUAGES)

- Definiscono sintassi e semantica dei messaggi, scambiati tramite Low Level Protocols
- Non si scambiano solo **FACTS** ma anche **PROPOSITIONAL ATTITUDES** (Believing, Feeling, Hoping, ...)

SPEECH ACT THEORY

- LOCATION** contesto del msg
- ILLOCUTION** intenzioni comunicate nel msg
- PERLOCUTION** conseguenze del msg.

- ACL's msg contains: { Sender, Receiver, Performatives, Content, Ontology }

La ONTOLOGY → Vocabolario/Accordi comuni per interpretare il **CONTENT**

Principi:

- Indipendenti dal protocollo di comunicazione
- Indipendenti dal linguaggio con cui l'agente è scritto
- Indipendenti dal linguaggio del **Content**, e dalla **Ontology** (se c'è)

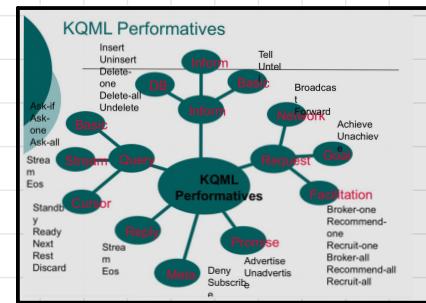
KQML

- Performatives
- Tell (P) : "Io credo P"
- Untell (P) : "Io non credo P"
- Insert (P) : "Sai che io crede P anche tu"
- ...

Attivatore

SENDER, RECEIVER, LANGUAGE, ...

• sono performativi per fare agenti **FACILITATOR** e **BROKER** che fanno da Router tra gli altri agenti



FIPA

• MSGs Sono Azioni (COMMUNICATION ACTS)

- INFORMATION EXCHANGE: inform, query-ref, met-undertake
↳ ADVANCED: inform-if, confirm, discard,...

- TASK DELEGATION: request, agree, refuse, ...
↳ ADVANCED: request-whom, ...

- NEGOTIATION: CFP (call for proposal), propose, accept-proposal, ...

• (INTERAGAZIONI) PROTOCOLLS: definiscono conversazioni strutturate

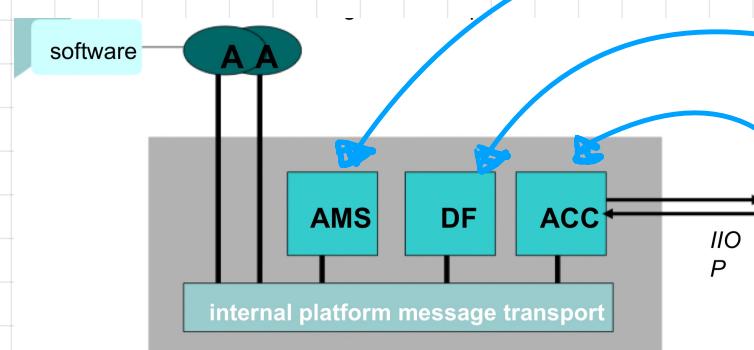
→ FIPA AGENT PLATFORM → guideline per implementazione

```
(request
:sender (:name
user_agent@bond.mchp.siemens.de:3410)
:receiver (:name
hilton_hotel@tcp://hilton.com:5001)
:ontology fipa-ptb
:language SL
:protocol fipa-request
:content
( action hilton_hotel@tcp://hilton.com:5001
( book-hotel (:arrival 04/07/1999)
(:departure 12/07/1999)
(:infos ( )) )
))
```

The **AMS** (Agent Management System) provides services like lifecycle management (creation, deletion, pausing, ...), name registration, name lookup, and authentication.

The **DF** (Directory Facilitator) provides yellow pages services which describe the attributes and capabilities of agents in the platform.

The **ACC** (Agent Communication Channel) accepts and delivers message between agents on different platforms (+store and forward, +firewalls)



KQML SEMANTICA

Preconditions indicate the necessary state for an agent in order to send a performative and for the receiver to accept it and successfully process it.

Postconditions describe the states of both interlocutors after the successful utterance of a performative (by the sender) and after the receipt and processing (but before a counter utterance) of a message (by the receiver).

Preconditions indicate what can be assumed to be the state of the interlocutors involved in an exchange. Similarly, the *postconditions* are taken to describe the states of the interlocutors assuming the successful performance of the communication primitive

FIPA SEMANTICA

A primitive's meaning is defined in terms of FPs and REs

The Feasibility Preconditions (FP) of a CA define the conditions that ought to be true before an agent may plan to execute the CA

The Rational Effect (RE) is the effect that an agent hopes to bring about by performing an action (but with no guarantee that the effect will be achieved)

The FPs and the REs involve agents state descriptions

TELL VS INFORM

The difference is only observable in the semantics

Syntactically the two messages in KQML and FIPA are almost identical

Both languages make the same basic assumption of non-commitment to a content language

Semantically they differ at two levels:

different ways to describe the primitive, i.e., pre-, post-, completion conditions for KQML, FPs and REs for FIPA ACL

different logic language to describe the propositional (mental) attitudes, e.g. belief

CONTRACT NET PROTOCOL \rightarrow c'è un Manager che distribuisce TASK

(1) Annuncia TASK agli ELIGIBLE AGENTS // MULTICAST

↳ **ELIGIBILITY SPECIFICATION**: CRITERI PER ESSERE ELIGIBILE

(2) Agli AGENTI rispondono con BID

↳ **BID DESCRIPTION**: DESCRIZIONE DEL FORMATO DEL BID

(3) Manager assegna contratto ad AGENTE (e apre canale di comunicazione privato) // PCP

Blindly committed agent

G - always
A - inevitable
E - optional

- maintains its intentions until it believes it has achieved them

$$\text{xInt(A Fp)} \rightarrow A (\text{xInt(A Fp)} \vee \text{xHelp}) \quad (\text{exclusive } \vee)$$

- an agent can be committed to means (**p** is an action) or to ends (**p** is a formula)

- defined only for intentions toward actions or conditions that are true for all paths in the agent's intention accessible worlds.

Single-minded committed agent

- maintains its intentions as long as it believes they are still options

$$\text{xInt(A Fp)} \rightarrow A (\text{xInt(A Fp)} \vee (\text{xHelp} \vee \neg \text{xBel}(E Fp)))$$

Open-minded committed agent

- maintains its intentions as long as these intentions are still its desires (goals)

$$\text{xInt(A Fp)} \rightarrow A (\text{xInt(A Fp)} \vee (\text{xHelp} \vee \neg \text{xDes}(E Fp)))$$

F - eventually
G - always
A - inevitable
E - optional

Semantics: Kripke Models

- A Kripke Model is a triple $M = \langle W, R, I \rangle$ where:
- W is a non empty set of **worlds**
- $R \subseteq W \times W$ is a binary relation called the **accessibility relation**
- I is an **interpretation function** $I: L \sqsubset \text{pow}(W)$ such that to each proposition P we associate a set of possible worlds $I(P)$ in which P holds

TEMPORAL LOGIC: Usata per definire formalismi logici per la specifica di comportamenti di sistemi che evolvono nel tempo

- The time may be **linear** or **branching**; the branching can be in the past, in the future or both
- Time is viewed as a **set of moments** with a strict partial order, $<$, which denotes temporal precedence.
- Every moment is associated with a possible state of the world, identified by the propositions that hold at that moment

Modal operators of temporal logic (linear) LTL

$p \mathbf{U} q$ - p is true until q becomes true - **until**

$\mathbf{X} p$ - p is true in the next moment - **next**

$\mathbf{P} p$ - p was true in a past moment - **past**

Typical temporal operators used are

$\bigcirc \varphi$	φ is true in the <i>next</i> moment in time
$\Box \varphi$	φ is true in <i>all</i> future moments
$\Diamond \varphi$	φ is true in <i>some</i> future moment
$\varphi \mathbf{U} \psi$	φ is true <i>until</i> ψ is true



$\mathbf{F} p$ - p will finally (eventually) be true in the future - **eventually**

$\mathbf{G} p$ - p will always be true in the future – **always**

BRANCHING

- Temporal structure with a branching time future and a single past - **time tree T**
- CTL – Computational Tree Logic**
- In a branching logic of time, a **path** at a given moment is any maximal set of moments containing the given moment and all the moments in the future along some particular branch of **T** (**so, a path is actually a subtree**)
 - * **Situation** - a world w at a particular time point t , wt
- State formulas** - evaluated at a specific **time point** in a time tree
- Path formulas** - evaluated over a **specific path** in a time tree

Modal operators over path formulas (branching)

$\mathbf{A} p$ - at a particular time moment, p is true in all paths emanating from that point - **inevitable p**

$\mathbf{E} p$ - at a particular time moment, p is true in some path emanating from that point - **optional p**

DALI: TELL/KNOW

- TOLD : $\{ \text{f1} \text{f2} \} \rightarrow \boxed{\text{msg}} \text{ sono ricevuti solo se soddisfanno delle condizioni}$
- TELL : $\{ \text{f1} \text{f2} \} \rightarrow \boxed{\text{msg}} \text{ sono inviati} \quad " \quad " \quad " \quad "$

$\rightarrow \text{Primitive}(\text{Content}, \text{Sender})$

$\rightarrow \text{TOLD}(\text{Sender}, \text{primitive}(\text{Content})) :- \text{const}_1, \dots, \text{const}_n$

$\rightarrow \text{TELL}(\text{Receiver}, \text{Sender}, \text{primitive}(\text{Content})) :- \text{const}_1, \dots, \text{const}_n$

5 Meta-reasoning layer OPTODOLE DALI

In heterogeneous Multi-agent Systems, in general not all the components speak the same language, and not all of them use the same words to express a concept. The agent that doesn't understand a proposition can either accept the defeat and ignore the message, or try to apply a reasoning process in order to interpret the message contents. The latter solution can be more easily put at work by taking advantage of meta-reasoning capabilities of a logic language. In fact, the use of *ontologies*, which are dictionaries of equivalent terms, can be integrated with several kinds of commonsense reasoning. The ontology of a DALI agent is in a file .txt containing equivalent terms and other properties useful in the meta-reasoning process. E.g., agent *bob*'s ontology can be the following, where *symmetric* is a property of relations, which is asserted to hold of predicate *friend*, and allows him to conclude both *friend(bob,lucy)* and *friend(lucy,bob)* even if originally he could derive only one. The name of the agent enclosed to each item of the ontology allows a group of agents to use the same ontology file, though sharing the contents only partially.

```
ontology(bob, rain, water_falling_from_sky).
ontology(bob, friend, amico).
... symmetric(friend).
```

APPROXIMATE

(1) ABSTRACT

→ Purely Reactive (Non considerano lo Stato)

con Stato

(2) PRACTICAL REASONING: consiste di 2 attività:

1. DESIDERAZIONE decidere l'obiettivo → INTENZIONI

2. MEANS-END REASONING: decidere come raggiungere l'obiettivo → PIANI